2017

Harmful Algae Bloom Prediction Model for Western Lake Erie Using Stepwise Multiple Regression and Genetic Programming

Amin Daghighi
Cleveland State University

HARMFUL ALGAE BLOOM PREDICTION MODEL FOR WESTERN LAKE ERIE USING STEPWISE MULTIPLE REGRESSION AND GENETIC PROGRAMMING

AMIN DAGHIGHI

Bachelor of Science in Civil Engineering

University of Tehran

19th February 2015

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN CIVIL ENGINEERING

at the

CLEVELAND STATE UNIVERSITY

July 2017

We hereby approve this thesis for

Amin Daghighi

Candidate for the Master of Science in Civil Engineering degree for the

Department of Civil and Environmental Engineering

and the CLEVELAND STATE UNIVERSITY

College of Graduate Studies

Thesis Chairperson, Dr. Ungtae Kim

Department & Date

Thesis Committee Member, Dr. Mehdi Jalalpour

Department & Date

Thesis Committee Member, Dr. Walter M Kocher

Department & Date

Student's Date of Defense: July 26, 2017

ACKNOWLEDGMENTS

I would like to thank my thesis committee and Dr. Edward J. Magiste, for all of their suggestions and guidance during the writing of this thesis and during my graduate study. Without them, this thesis would not have been possible. I want to thank my supportive major advisor Dr. Ungtae Kim for his guidance, support and numerous suggestions, especially in data selection algorithms and for improving the quality of this thesis. He was supporting me financially for my master program for two years. I want to give my thanks to my Father and my Mother for supporting me from Iran, without their warmly supports I was not able to live here in peace, and a great especial thanks to my lovely younger brother Ali Daghighi, for being an awesome brother and his supports all the time. A special thanks to my friends especially Nick and Seema for working alongside me and supporting me these last two years.

HARMFUL ALGAE BLOOM PREDICTION MODEL FOR WESTERN LAKE ERIE USING STEPWISE MULTIPLE REGRESSION AND GENETIC PROGRAMMING

AMIN DAGHIGHI

ABSTRACT

The Great Lakes are most important freshwater bodies providing water resources and other various related businesses to the northeastern part of North America. However, harmful algal blooms (HABs) are more often and severe in those lakes than before and thus threatening lake environments and economies. Researchers have studied the factors influencing HABs characteristics using different scientific methods. In this study, all possible predictors and predictand variables were collected from various data source and then eight final predictors and one predictand were selected based on correlation between predictors and predictand variables. This study tests two machine learning techniques, Stepwise Multiple Regression (SMR) and Genetic Programming (GP), to forecast monthly HAB indicators in Western Lake Erie from July to October. SMR and GP models were created with selected input variables for two training periods, 2002 to 2011 and 2002 to 2014. A Spearman rank correlation coefficient was used to choose input variable sets for each HAB month considering 224 different combinations of lag time and average periods. The SMR models showed a correlation coefficient increase from 0.71 to 0.78 when extending the training period. The GP models followed a similar trend

increasing the overall correlation coefficient from 0.82 to 0.96. Both models optimally selected monthly discharge and phosphorus mass from Maumee River Basin as significant predictor variables. A major drawback of both models was data-dependency as common in data-driven methods. GP was better to detect high nonlinear HAB mechanism than SMR due to its nature to use many mathematical functions while SMR only use the linear combination of variables. This study attested that both SMR and GP can be useful to simulate historical HAB event and predict future HAB severity. In future work, to avoid under- or over-prediction for unobserved HAB mechanism regarding short training period, it is suggested to develop an extrapolation technique that is statistically sound and operable in the model and test multi-model ensemble approaches to provide most possible HAB prediction.

Key Words: Harmful Algal Blooms, Genetic Programming, Stepwise Multiple Regression, Data-driven Methods.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

HAB	Harmful Algal Bloom
Air	Air Temperature
Chl-a	Chlorophyll-a
CI	Cyanobacterial Index
EPA	Environmental Protection Agency
GLERL	Great Lakes Environmental Research Laboratory
GLM	Great Lakes Monitoring
GLNPO	Great Lakes National Program Office
GP	Genetic Programming
NCWQR	National Center for Water Quality Research
NOAA	National Oceanic and Atmospheric Administration
PM	Phosphorus Mass
ppb	Parts Per Billion
Q	Flow Rate
SMR	Stepwise Multiple Regression
SRP	Soluble Reactive Phosphorus
TKN	Total Kjeldahl Nitrogen
TP	Total Phosphorus
trainbr	Bayesian Regularization Backpropagation
USGS	United States Geological Survey
Water	Water Temperature
Wind	Wind Speed
WLEEM	Western Lake Erie Ecosystem Model

CHAPTER I

INTRODUCTION

1.1 Problem Statement

Harmful Algal Blooms (HABs) are quickly becoming a key problem all around the world. A HAB is a bloom of algae that has the potential to harm humans or the ecosystem (Ho, & Michalak, 2015) The current state of knowledge regarding HABs, their growth, and means of addressing the issues resulting from them, stems from a rich literature on the taxonomy, growth characteristics, and ecophysiology of freshwater and marine phytoplankton collectively grouped as “harmful algae.”

This societally defined category includes toxic species that express poisonous substances to higher trophic levels, largely fish, shellfish, marine mammals, or humans, and include members of the cyanobacteria, dinoflagellates, raphidophytes, haptophytes, and diatoms. Included also under the HAB umbrella are largely human-caused high-biomass events that, while often comprising non-toxic phytoplankton species, still critically alter ecosystems through hypoxia/anoxia, altered food web efficiencies, stimulation of pathogenic bacteria, or other ecological consequences. (Wells et al., 2015)

HABs are pervasive along coastlines of most nations including the United States (Hallegraeff, 1993, Anderson et al., 2008). Excessive nutrient loading is commonly cited as a factor contributing to the expansion of HAB (Anderson, Glibert, & Burkholder, 2002, Heisler et al., 2008). However, zooplankton grazing plays an important role in constraining phytoplankton abundance in aquatic ecosystems (Burkill, Mantoura, Llewellyn, & Owens, 1987, Latasa, Landry, Louise, & Bidigare, 1997, Calbet, & Landry, 2004) and a failure of predator control can facilitate phytoplankton blooms (Irigoien, Flynn, & Harris, 2005, Modigh, & Franzè, 2009). Zooplankton grazing has also been shown to have a primary effect on the outbreak of HAB (Smayda, 2008), and for some HAB such as caused by the pelagophyte, Aureoumbra lagunensis, blooms may be promoted via positive feedback between grazing disruption and altered nutrient cycling (Kang, Koch, & Gobler, 2015).

HABs in freshwater systems are quickly becoming a global epidemic as well. Reports of HABs in Lake Taihu in China (e.g., Qin et al., 2010), Lake Victoria in Africa (e.g., Sitoki, Kurmayer, & Rott, 2012), Lake Erie in North America (e.g., Michalak et al., 2013), and Lake Nieuwe Meer in The Netherlands (e.g., Joehnk, Huisman, Sharples, Sommeijer, Visser, & Stroom, 2008) constitute examples of an alarming trend in freshwater ecosystems worldwide that is only expected to worsen under a changing climate (Paerl, & Huisman, 2009). The effects of HABs are well documented. These effects are associated with acute morbidity and mortality across a range of biota (including humans) (Landsberg, 2002, Van Dolah, 2005), economic impacts through ecological and human health costs (Anderson, Hoagland, Kaoru, & White, 2000, Hoagland, Anderson, Kaoru, & White, 2002) and the need for additional water treatment

measures for regions relying on surface water supplies (Hitzfeld, Höger, & Dietrich, 2000, Hoeger, Hitzfeld, & Dietrich, 2005).

HABs in Great Lakes is becoming a major issue recently more than last decade. The Laurentian Great Lakes, so named because of their relationship to the St. Lawrence River, are arguably one of the most valuable natural resources in North America, if not the world. This system situated between Canada and the Mid-western United States represents roughly 20% of the earth's available surface freshwater, a resource that is expected to become increasingly limited in the near future (Schottler, Eisenreich, & Capel, 1994). Lake Erie alone provides over 7 billion dollars in revenue each year from tourism and fishery industries (United States Department of Agriculture (USDA), 2005).

Among the five Great Lakes, Lake Erie is most susceptible to recurring large-scale blooms due to the morphology of the lake, its location in a temperate climate with warm summer temperatures, and extensive anthropogenic inputs. At an average depth of 19 meters, Lake Erie has a relatively short retention time (less than 3 years) and consistently reaches temperatures above 25 °C during summer months (Stumpf, Wynne, Baker, & Fahnenstiel, 2012).

Lake Erie is the cause for concern regarding threats to the fresh water system. For example, an HAB in Lake Erie during the summer of 2014 resulted in a three-day tap water ban for Toledo, Ohio (Wilson, Wright, Bronnenhuber, MacDonald, Belore, & Locke, 2014), providing an acute reminder of the impacts of HABs and the urgency of addressing their proliferation. The need for scientifically guided policy to mitigate these impacts has never been greater; since the 1990s, harmful Microcystis blooms reappeared

in Lake Erie and plankton growth has been enhanced in response to the increasing total phosphorus loading and weather-driven changes (Michalak et al., 2013).

1.2 Impacts of HABs

Current spatial and temporal ranges of HAB species will most certainly change under future climate scenarios. Spatially, one can expect that the geographic domains of species may expand, contract, or just shift latitudinally. The HAB research community is largely under-prepared to address these questions. The central challenge is to achieve consensus about the way forward from both research and management perspectives. This focused community synergy will be critical if the knowledge base is to advance faster than the influence of climate-related changes on HABs, and if statistically credible evidence of this change can be provided soon enough to contribute to the societal debate over climate change impacts. These preparations will be particularly critical for high latitude regions where climate change impacts are liable to be most rapid and substantial (Stocker et al., 2013).

The foundation of HAB knowledge has accumulated mainly through isolated investigations, as with most environmental sciences, but this piecemeal process does not readily foster as powerful a knowledge structure as can be achieved through synergistic, collective, and collaborative approaches. That is, a collective vision is needed that can identify the ‘‘known knowns’’ and rank the levels of the ‘‘known unknowns’’ if the community is to presage climate change-HAB linkages before they develop (Wells et al., 2015).

Another possible factor for the return of HABs is the zebra and quagga mussels. The mussels filter small particles out of the water such as algae, microscopic bugs, or

zooplankton that eat algae (Reutter, & Dierkes, 2014). They then excrete dissolved phosphorus, a main source of food for HABs. If the mussels suck in a harmful form of algae, they stop filtering and spit it out then start filtering again (Reutter, & Dierkes, 2014).

In working to achieve a higher level of cooperation among HAB and climate scientists, there is some guidance to be gleaned from the ocean acidification field, which used broad collaboration to create the infrastructure and standard methods needed to generate scientific awareness and funding streams that critically address the environmental and biological questions of greatest importance. Moving the understanding of HAB-climate change interactions beyond informed speculation will require rigorous, testable hypotheses to guide scientists, managers and the public on what changes are happening or are projected, estimation of the confidence limits on those projected changes, and establishing the infrastructure and studies needed to capture these necessary data.

HAB-climate change interactions directly affected by depth of the lake or coastal area. Lake Erie as the smallest and shallowest system of the Great Lakes, and therefore, the most susceptible to nutrient-driven water quality issues, is uniquely positioned to be a proving ground for the hypothesis expressed in the current study. Recent evidence suggests that rapid ecological changes are in fact occurring in the ecosystem, involving a complex and often poorly understood interplay among many factors related to the lake's chemical, physical and biological characteristics (Michalak et al., 2013). The problem of HABs has been becoming an increasingly larger problem for Western Lake Erie as

shown in Figure 1. The 2011 bloom peak was 274% larger over the previous peak bloom of the previous nine years.

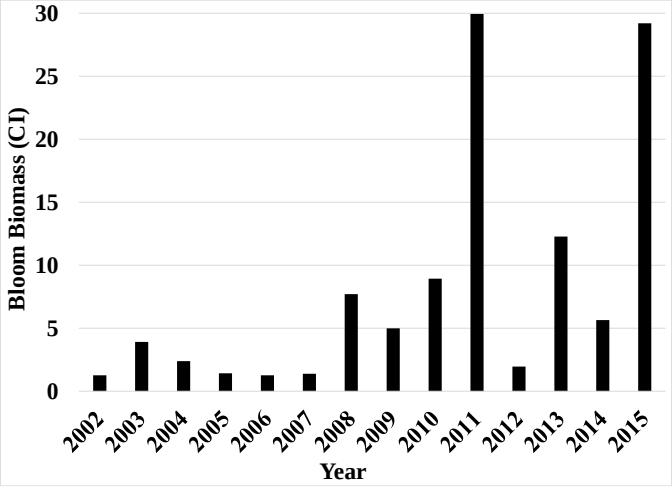


Figure 1. Time Series of HAB Peak Biomass at Lake Erie from 2002~2015

Recently Western Lake Erie faced one serious calamity. On August 2, 2014, the City of Toledo's water treatment plant was shut down until August 4th. The bloom was not large in terms of coverage throughout the lake however the bloom was very thick and happened to be concentrated where the water treatment plant's intake pipes are located. When the water in Lake Erie was tested the Microcystin toxin levels were between ten to twenty parts per billion (ppb) (Kozacek, August 2014).

The World Health Organization has set the following guidelines for Microcystin in Ohio: children under six and sensitive populations do not drink when the toxin levels reach 0.3 ppb, ages six and older when there is a concentration level of 1.6 ppb, and when the toxin levels reach twenty ppb the water should not be used (Environmental Protection Agency, 2017). The drinking water crisis left more than four hundred thousand people and three counties in Ohio and one in Michigan without drinking water. The governor of

Ohio, John Kasich, announced a state of emergency to organize resources for the affected (Kozacek, August 2014). Humanitarian organizations like the American Red Cross responded by manning water distribution centers and provided water delivery assistance to homebound residents thus indicating that HABs can have serious effects on local economies.

HABs can have a major effect on property values in Western Lake Erie as well. A study performed to look at the economic effects of HABs determined there is 3.458 billion dollars in residential housing stock near the western basin of Lake Erie (Bingham, Sinha, & Lupi, 2015). Recreational activities such as boating, water skiing, fishing, and swimming are all effected when HABs occur. Water treatment plants must take more precautions and use more treatment methods in an attempt to not repeat what happened in Toledo in 2014. Tourism is also an important economic factor; millions of trips are taken to counties near Western Lake Erie with a range of sixty-six million to three hundred and five million dollars at risk (Bingham, Sinha, & Lupi, 2015). Table 1 shows the result of the study on economic loses from the 2011 and 2014 HABs.

Table 1. Breakdown of HAB Effects on the Ohio Economic Losses

Economic Factors	HAB Event Year	
	2011	2014
Property Value	$16,000,000	$18,000,000
Tourism	$20,000,000	$20,000,000
Recreation	$31,000,000	$23,000,000
Water Treatment	$4,000,000	$4,000,000
Overall	$71,000,000	$65,000,000

HABs have the possibility of causing many different types of health problems for humans and animals as well as having major effects on the economy. The most common

species of harmful algae in Ohio's Great Lake is Cyanobacteria also known as blue-green algae. The Ohio Department of Health listed the health problems that go along with each type of exposure listed below (Ohio Department of Health, 2016):

1) Drinking or swallowing water contaminated with Cyanobacteria
    - Severe diarrhea and vomiting
    - Difficulty breath
    - Neurotoxicity (weakness, tingly fingers, numbness, dizziness)
    - Death
2) Skin Contact often from recreation activities in HAB waters
    - Rashes
    - Hives
    - Skin blisters
3) Inhaling water droplets of mists of Cyanobacteria contaminated water
    - Runny eyes and nose
    - Sore throat
    - Asthma-like symptoms

The species of HABs in Western Lake Erie is Microcystis where bloom growth is promoted by warm temperatures over twenty degrees Celsius. The months that consistently have temperatures over the temperature threshold are July, August, and September. The months that often have blooms are the three months over the temperature threshold with a carry over into October. HABs are being forecasted by different techniques around the world.

A variety of aquatic biogeochemical models have been developed to understand ecological interactions and to predict the response of Lake Erie to external nutrient loading changes. Some of the models were constructed during the mid-1970s (e.g., Di Toro, Thomas, Herdendorf, Winfield, & Connolly, 1987, Scavia et al., 2014, Lam,

Schertzer, & Fraser, 1987) whilst a new generation of models has been in place more recently (e.g., Leon et al., 2011, Zhang, Culver, & Boegman, 2008). During the 2000s, HABs had returned to being a yearly problem for the Western Lake Erie basin.

Machine learning techniques have been increasingly used to forecast HABs. Dissimilar to traditional methods, machine learning is based on algorithms that are able to iteratively learn from data finding hidden insights without depending on rule-based programming. Supervised learning algorithms are often used when historical data is able to predict future events.

Forecasting HABs in Lake Erie will allow commercial as well as recreational users of the lake to make timely decisions concerning Western Lake Erie. There are two available HAB forecasting models for Western Lake Erie from the National Oceanic and Atmospheric Administration (NOAA). One of the forecasts is an assembly of multiple models to forecast the peak bloom for the year. The second forecast is focused on weekly short-term forecasting and provides size as well as location. The focus of this study is to bridge the gap between the two available forecasts.

Conventional, modeling of phytoplankton dynamics has been carried out using process-based models by incorporating physical and biotic environmental variables into a water quality model. However, this approach is reported to suffer from the uncertainty of kinetic coefficients used in such models. In the recent past, many studies have reported the successful application of data-driven Artificial Intelligence-based techniques, particularly the Stepwise Multiple Regression (SMR) and Genetic Programming (GP) techniques.

In this study, the main purpose is to suggest a systematic method to select significant input variables for HAB prediction for recognizing the most appropriate SMR and GP models from a list of possible models through a physical understanding of the HAB processes supported by data interpretation.

1.3 Research Objectives

The main objective of this thesis is to develop two data-driven HAB prediction models, SMR and GP, to improve operability and accuracy for monthly predicting the HAB at Western Lake Erie, compare them and confirm the applicability of the methods for other different lakes around the world. Specific research objectives are as follows.

1) Perform literature review to analyze the state-of-the-art in HAB prediction
2) Collect and document relevant predictor and predictand variables for Western Lake Erie
3) Systematically select predictor variables using statistical methods
4) Develop, train, and test SMR and GP models for HAB prediction in Western Lake Erie
5) Provide future research direction to apply the developed methods to other HAB prediction studies.

CHAPTER II

LITERATURE REVIEW

This study concentrates on forecasting harmful algae blooms (HABs) in the western basin of Lake Erie. The western basin of this lake has had problems with HABs for decades, however the central and eastern basins have not typically experienced large HABs. In addition to discussing the causes of HABs in Lake Erie's western basin, this study will explore the collection of data in other lakes around the world. Specifically, this study will examine the advantages and disadvantages of four machine learning techniques: Genetic Algorithm and Programming (GA and GP), Stepwise Multiple Regression (SMR), Artificial Neural Network (ANN), and classification and regression tree (CART). Furthermore, a review of the current literature concludes with an examination of several HAB forecasting models currently available for Lake Erie.

2.1 Environmental Variables of HABs

2.1.1 Study Area

There are two major factors which cause harmful algae blooms in lakes such as Erie: water depth and nutrient loading (Kim, Zhang, Watson, & Arhonditsis, 2014). The

average water depths for the eastern, central, and western basins are 24, 18.3, and 7.4 meters, in that order (Ohio Department of Natural Resources, 2017). The shallow waters in the western basin cause an increase in water temperature promoting the growth of HABs. This is correlated with the seasonal changes in northeast Ohio; spring and summer produce more blooms because there is more sunlight. The second major factor that promotes bloom growth in the western basin is nutrient loading from tributaries. There are two major tributaries that flow into Lake Erie. They are the Detroit River and the Maumee River. These each contribute the two main nutrients for Microcystis, which is HABs to bloom: phosphorus and nitrogen.

Both nitrogen and phosphorus have an effect on the HABs, however, without phosphorus there is little effect on the blooms. Thus, it can be said, that phosphorus is a limiting factor, in that, when it is present, even alone, it increases HABs.

River Flow Impact. The amount of flow from the Maumee River into Lake Erie is $1/35^{th}$ of the Detroit River, however the concentration of nutrients from the Maumee results in the same amount of nutrients as the Detroit River entering the lake (Stumpf, Johnson, Wynne, & Baker, 2016). The remaining tributaries—of which there are 11 water stations used for data collection – produce insignificant amounts (less than ten percent) of the nutrient loads, compared with the Maumee River (Stumpf et al., 2016). Preliminary analysis of the data indicates that the nutrient loads from the Maumee River are the main source of nutrients for HABs. Albeit that, the Maumee is smaller in flow, the amount of nutrient loading is the same as the Detroit, which indicates that the Maumee's nutrient loading is more concentrated than the Detroit River (Wayne & Stumph, 2015).

Western Basin Data Location. The collection site for this study is shown Figure 2. Great Lakes Monitoring (GLM) collection sites are shown with green and blue points on map. The Environmental Protection Agency (EPA), and Great Lakes National Program Office (GLNPO) collection sites are shown with purple and orange points on map. All stations, including river stations, controlling stations, and all buoys, in Western Lake Erie are marked in the Figure 2, and shown below.

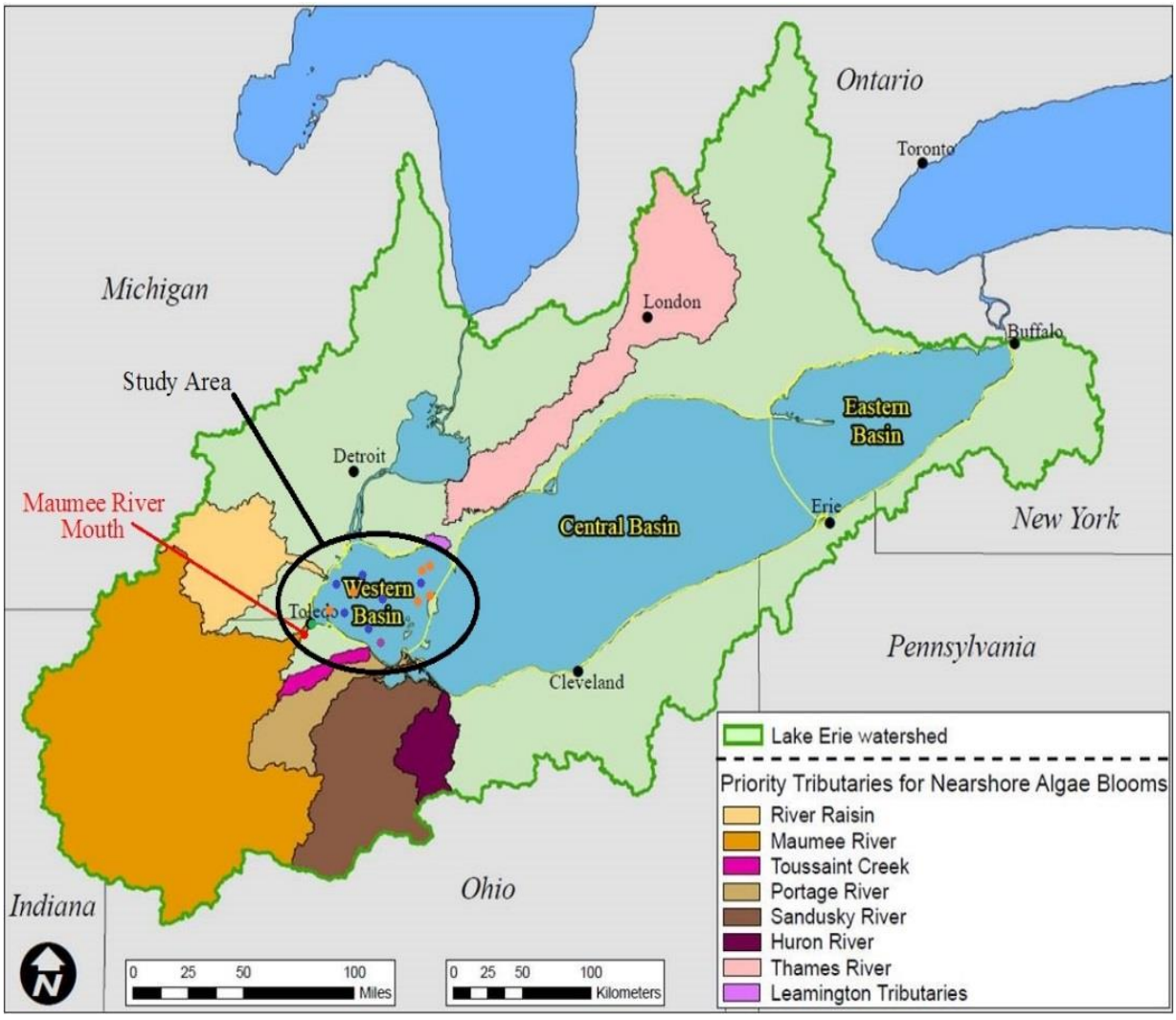


Figure 2. Western Lake Erie Study Area and Data Collectors' Site (Environmental Protection Agency [EPA], 2017)

2.1.2 Prediction Variables of HABs

The most important phase, and the first step of modeling the algae bloom index, is finding the correct and correlated variables as the predictors, and a suitable predictand variable as a HAB index. Traditionally, researchers have looked at both nutrient loading and climate factors to formulate models predicting HABs. Often, these previous studies have focused on either nutrient loading aspects or weather aspects rather than using a combination of climate, nutrition, and watershed characteristics. Yet the focus on one set of variables as the basis for prediction has been limited, this creates a gap in proper understanding and predicting HABS.

Kang, Koch, and Gobler, (2015), selected nitrate, nitrite, ammonium, phosphate, silicate, urea, total dissolved nitrogen, and dissolved organic nitrogen. Based on literature review in the area, and nutrient amendment experiments were conducted to assess how specific types of nutrient loading would affect the growth of multiple groups of phytoplankton during brown tide blooms in Florida (Kang et al., 2015). One-way analyses of variance with a post-hoc Tukey test were performed to assess statistically significant differences, while G-tests were used to compare frequencies of experiments in which grazing rates were measurable on major eukaryotic and prokaryotic populations.

Changes in macronutrient supply and form will lead directly to a switch towards HAB species and bloom events in most marine environments, in contrast to the impact of increased phosphorus and nitrogen inputs to brackish and freshwater environments (e.g., Thornton et al., 2013).

Some studies focused on weather conditions rather than nutrient. 17 weather condition factors and 12 water quality factors, including Chlorophyll-a (Chl-a), monitored monthly

to analyze cyanobacteria bloom in the Waihai part of Dianchi Lake, China. A probabilistic non-linear regression use for predicting the probability of occurrence of a 0-1 event by fitting data to a logit function, so that it recognizes the impact of dependent variables that may be either numerical or categorical (Sheng, Liu, Wang, Guo, Liu, & Yang, 2012).

Lou, Xie, Ung, & Mok (2015), chose 23 water quality parameters, including hydrological, physical, chemical and biological parameters, were monitored monthly and these 23 parameters were measured according to the standard methods. In order to identify the water parameters that were significantly correlated with phytoplankton abundance, correlation analysis, Particle Swarm Optimization - Support Vector Regression (PSO–SVR), was conducted firstly. The forecast model was based on the last three months data. Including the three-month data in this forecast model is to adopt the historical effect of the last year that have similar environmental conditions such as temperature influence the growth of phytoplankton (Lou et al., 2015).

In Rajaee, & Boroumand (2015) paper, discrete wavelet transform (DWT) with artificial neural network (ANN), multi linear regression (MLR), and genetic algorithm-support vector regression (GA-SVR) models were developed for one month ahead prediction of eutrophication in San Francisco Bay gauging station in the USA, and were compared together. To achieve the best combination of input data driven from time series, two statistical measures of goodness of fit the Nash–Sutcliffe model efficiency coefficient (E) and root-mean-squared error (RMSE) between Chl-a_{t+1} with Chl-a_t, Chl-a_{t-1}, Chl-a_{t-2}, …, Chl-a_{t-i} time series were computed and presented. The combined data with 3, 6 and 12 months delay was used to investigating the effects of seasonal variation

of the Chl-a on the value of Chl-a_{t+1} and the combination with highest R^2 selected as input.

In short, historically various combinations of variables have been combined to produce HAB models. These are summarized in Table 2, and presented here below.

Table 2. HAB Predictor Variables Selected in Other Studies

Study	Model	Variables
Lou et al. 2015	Hybrid intelligent model	1,3,4,5,6,7,8
Wu et al. 2015	Satellite image acquisition and analysis	2, 9
Rajaee, & Boroumand, 2015	Artificial Neural Network (ANN) and Multi Linear Regression (MLR), and Genetic Algorithm-Support Vector Regression (GA-SVR)	9
Cho et al. 2014	Artificial Neural Network (ANN)	1,3,5,6,7
Persaud et al. 2015	Time series analyses and Pearson pairwise correlations	1,2,3,6,7
Maier et al. 2001	Artificial Neural Network (ANN)	1,3,6,7,8
Håkanson et al. 2003	Coefficients of variation	9
Qin et al. 2015	The monitoring comprised three different elements: (i) remote sensing image retrieval (ii) unattended sensor detection, with wireless data transmission (iii) ship-borne sampling and analysis	2,3,9
Talib et al. 2008	Artificial Neural Network (ANN) and Hybrid Evolutionary Algorithm	1,6,7,9
Kim et al. 2014-a	Three multiple modelling approaches	1,6,7,9
Bertani et al. 2016	Bayesian HAB model	7
Jia et al. 2013	Statistical analysis	1
Zhang et al. 2013	Windows-based Software integrating the EcoTaihu model	6,7,9
Kim et al. 2014-b	The Takagi-Sugeno fuzzy model, and Discrete wavelet transform algorithms	1,3,5,6,7
Feng et al. 2015	Two-dimensional mathematical model	1,2,9
Chen et al. 2015	Auto-Regressive Integrated Moving Average (ARIMA)	1,6,7,9
Stumpf et al., 2016	Non-linear relationships	5, 7, 10
1-Temperature 2-Wind Speed 3-Precipitation 4-Alkalinity 5-Dissolved Oxygen 6-Nitrate 7-Phosphorus 8-Turbidity 9-Chl-a 10-Cyanobacteria Index		

2.2 Variables Selected for Current Study

After considering all the potential accepted variables evidenced in the literature, searching through several databases, and data gathered from Heidelberg University's National Center for Water Quality Research (NCWQR) (National Center for Water Quality Research, 2017), United States Geological Survey (USGS) (United States Geological Survey government agency, 2017)

One main objective of this study was to gather all possible predictor and predictand variables to be considered. Initially, there are a total of twenty predictor and four predictand variables considered show in Table 3 and Table 4.

In Table 3 the first eight variables were obtained from the Heidelberg Tributary Loading Program operated by Heidelberg University's National Center for Water Quality Research (NCWQR). Water samples were taken on the Maumee River at Waterville, OH at the United States Geological Survey (USGS) station (04193500), one to three samples are analyzed a day depending on times of high flow or turbidity. The ten variables were taken from Great Lakes Monitoring (GLM) from the Illinois-Indiana Sea Grant (Great Lakes Monitoring, 2017). The Cyanobacterial Index (CI) data was gathered from NOAA and Stumpf (2016). Chl-a data was taken from GLM and EPA's Great Lakes National Program Office (GLNPO). Two satellites were used: the Medium Resolution Imaging Spectrometer for 2002-2011 and the Moderate Resolution Imaging Spectroradiometer for 2012-2015 (Stumpf, 2016). Ten-day composite images of the maximum CI at each map pixel were determined by using the satellite images to determine the total biomass for the ten-day periods from July 11th to October 31st (Stumpf, 2016). After collecting these ten-day CI values, the values were converted to the max value of the month as well as the

average of the CI values in each month. After finalizing all the possible inputs in Chapter II, through the Chapter II with statistical methods final inputs for all methods are selected.

Table 3. Collected Predictors Variables for Western Lake Erie

Variable		Source	Data Preiod	Update
Q	Flow Rate (cfs)	NCWQR	Jan 1975 Oct 2016	Daily
SS	Suspended Solids (mg/L)	NCWQR		
TP	Total Phosphorus (mg/L)	NCWQR		
SRP	Soluble Reactive Phosphorus (mg/L)	NCWQR		
TKN	Total Kjeldahl Nitrogen (mg/L)	NCWQR		
CL	Chloride (mg/L)	NCWQR		
Su	Sulfate (mg/L)	NCWQR		
Si	Silica (mg/L)	NCWQR		
Q	River Discharge (cms)	USGS	Oct 1966 Oct 2016	
TP	Phosphorus Load (mg/L)	GLM	Jan 2001 Jan 2017	
SRP	Soluble Reactive Phosphorus (mg/L)	GLM		
N	Nitrogen Load (mg/L)	GLM		
T	Turbidity (NTU)	GLM	April 1983 - August 2012	Each Apr and Aug Sporadic
A	Alkalinity (mg/L)	GLM		
N-N	Nitrite-Nitrate (mg/L)	GLM		
TP	Total Phosphorus (ug/L)	GLM		
DO	Dissolved Oxygen (mg/L)	GLM		
Water	Water Temperature (C)	USGS	January 2001 - February 2017	
Air	Air Temperature (C)	USGS		Daily
Wind	Wind Speed (knots)	USGS		

Table 4. Collected Predictand Variables for Western Lake Erie

Variable		Source	Data Period	Update
Chl-a	Chlorophyll a (ug/L)	GLM	April 1983 - August 2012	Each Apr and Aug
Chl-a	Chlorophyll a (ug/L)	EPA GLNPO	April 1999 - October 2011	Sporadic
CI	Cyanobacteria Index	NOAA	Jul 2002 - October 2015	Each 10 days
Mc	Microcystin (ug/L)	EPA GLNPO	January 1977 - December 2012	Sporadic

2.3 Harmful Algal Bloom Modeling

Since the 1990s, machine learning has been used to solve many complicated problems in various fields. Machine learning is an area of computer science and a sub-area of artificial intelligence concentrating on theoretical foundations (Muttil & Chau, 2006). Machine learning, in general, contains algorithms that estimate dependency between a systems inputs and outputs while improving its performance automatically through a training period. These different methods are then able to predict outputs from given inputs. These techniques are ideally suited to model the HAB dynamics since such models can be set up rapidly and are known to be effective in handling dynamic, non-linear and noisy data, especially when underlying physical relationships are not fully understood, or when the required input data needed to drive the process-based models are not available (Muttil & Chau, 2006). Three artificial intelligence algorithms are examined in this literature review: Artificial Neural Network (ANN), Stepwise Multiple Regression (SMR), and Genetic Programming (GP). The strengths of each algorithm were evaluate using three criteria as shown in Figure 3 (Kim, 2009):

- Knowledge Engineering Function which is the process of acquiring knowledge and refining it to gain additional knowledge

• Problem solving such as scheduling and optimization

• Classification & prediction

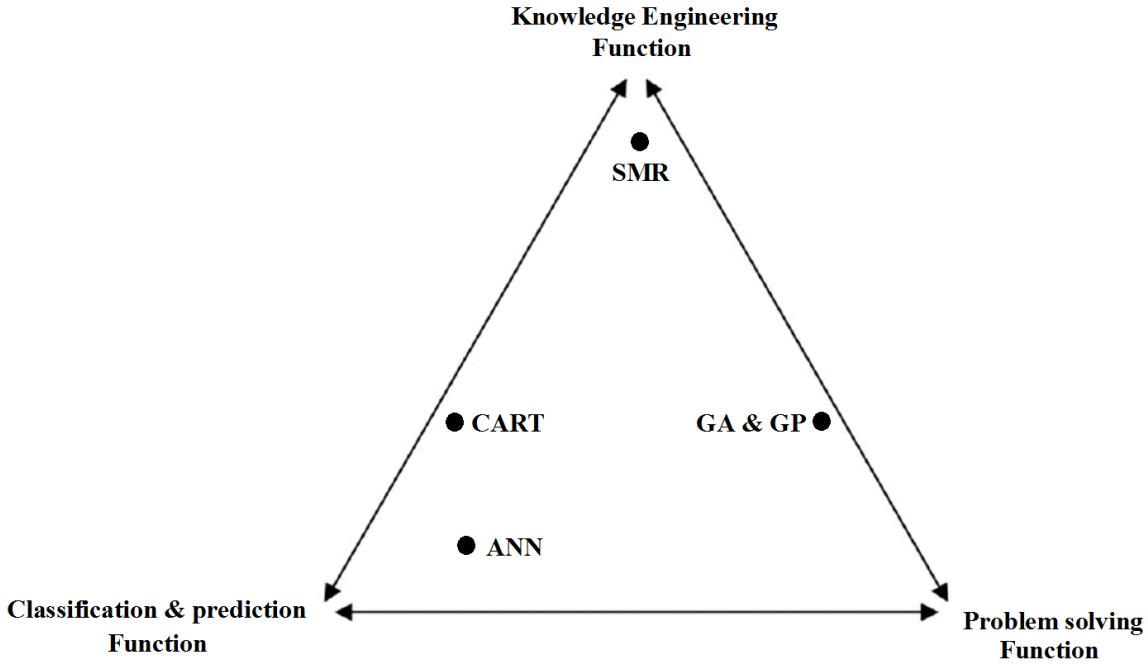


Figure 3. Strengths of Each of the Three Algorithms for Three Major Tasks

As early as in 1997, Recknagel et al. (1997) demonstrated that ANN is capable of modeling the non-linear and complex algal growth phenomenon. Lee et al. (2003) found that the algal concentration in samples from Tolo Harbor is primarily dependent on their antecedent concentrations in the immediately preceding weeks, and this result was supported by interpretation of the neural network weights.

2.3.1 Genetic Algorithm and Programming Method

The algal bloom phenomenon (particularly the red tide) has been widely reported and has become a serious environmental problem due to its adverse influence on aquatic life and on human health. The need for a better understanding of harmful algal bloom (HAB)

dynamics and the complex ecological processes involved in blooms is clearly evident from years of research (Lee and Qu, 2004). In spite of the extensive research that has already been undertaken, the causality and dynamics of algal blooms are not well-understood, and the prediction of algal blooms remains a very difficult problem due to the extremely complicated ecological dynamics of these systems (Taranu, Gregory-Eaves, Steele, Beaulieu & Legendre, 2017). Thus, it is highly desirable to obtain mathematical models that can give some insight into the physical properties of this process while having the capability to predict the occurrence of algal blooms with an acceptable degree of accuracy and lead-time (Michalski, Carbonell & Mitchell, 2013). Machine learning has been used to explain many different complicated problems since the 1990s. Machine learning is an area of computer science and a sub-area of artificial intelligence concentrating on the theoretical foundations (Muttil, 2006).

Muttil & Chau (2006) reported that both ANN and GP correctly identified the ecologically significant variables and that long term algal growth can be predicted using only chlorophyll-a as an input. They also observed that, when the 'maximum initial tree size' and 'maximum tree size' are restricted to 45 and 20, respectively, the evolved equation contains only 4–8 variables, and thus, the equation is easy to interpret.

Whigham & Recknagel (1999) compared GP-evolved equations with ANN models to demonstrate the applicability of GP to nonlinear processes in natural systems such as freshwater systems. They concluded that the transparent nature of GP solutions may allow inferences about underlying processes to be made, and they highlighted issues with scaling data for machine learning and the difficulty involved with producing understandable models.

Recknagel, Bobbin, Whigham, & Wilson (2002) compared the potential of ANN and GA in terms of prediction and understanding algal blooms in Lake Kasumigaura, Japan and found that models evolved by GA perform better than ANN models. Muttil & Lee (2007) studied modeling of algal bloom with GP to relate hydrometeorological and water quality data in Hong Kong. Final analysis of the results of GP models demonstrate that GP is cable of finding the connection between the natural auto-regressive of bloom peak time and dynamics and identifying the important input variables properly, in accordance with ecological reasoning. This study demonstrates that GP can be a practicable alternative to model the HABs with analytical system of the developed equations.

The use of GP in forecasting HABs is not without its advantages and disadvantages. One disadvantage of using the GP algorithm is that the user must decide a number of parameters before applying the algorithm to model the data, such as number of equations and number of calculation generations. The main advantage of GP is its ability to produce models that build a definitive formula or equation (Whitley, 2014).

2.3.2 Stepwise Multiple Regression Method

SMR and principal components analysis have long been used to select descriptive variables for relating runoff to climate and watershed descriptors. Statistical prediction methods, on the other hand, rely on past historical data for prediction. Techniques such as regression analysis, time-series analysis and artificial intelligence analyze the historical dataset to forecast the algal bloom (Chang, Shen & Chen, 2004).

Many researchers used different methods to select the best variables and also find a fit model to predict a data set, commonly used methods include SMR and Linear Regression

(Heuvelmans, Muys & Feyen, 2006; Brandes, Hoffmann & Mangarillo, 2005; Barnett, Gray & Tootle, 2009; Gong, Wang, Condon, Shearman & Lall, 2010; Peña-Arancibia, J. L., Van Dijk, A. I. J. M., Mulligan, M., & Bruijnzeel, 2010). The main difference between for example conventional inventory-based models and the SMR method of modeling approach is that the SMR model is less complex and, such a model approach allows the simulation of different scenarios by varying the values of input variables. (Chen, Shi, Shu & Gao, 2013)

Common problems relating to the SMR model include its lower performance with respect to artificial intelligence techniques and its lack of ability to extend the response to non-central positions of explanatory variables (Ul-Saufie, Yahya, Ramli & Hamid, 2012; Sayegh, Munir & Habeebullah, 2014). However, it is still generally used due to its simplicity.

Thus, SMR has an advantage in avoid the collinearity, however, out of range events can be neglected. SMR is a type of multiple linear regression that can select the best-fitted combination of predictor variables for predictand variable prediction with forward-adding and backward deleting variables (Abdelmutalab, Assaleh & El-Tarhuni, 2016). The stepping procedure begins as an initial model definition, with a stepped forward addition of a variable to the previous model. The critical F value is then used to check the eligibility of the added variable (Sharma & Yu, 2015). With a new variable added, the previous variables in the model may lose their predictive ability. Thus, stepping criteria are used to check the significance of all the included variables. If the variable is insignificant, then the backward method is used to delete it. Forward adding and backward deleting are repeated until no variable is added or removed. The stepping

procedure is eliminated when the optimized model is established (Dudek, 2016; Darlington & Hayes, 2016; Faraway, 2016).

CHAPTER III

METHODS AND MATERIALS

This chapter presents the potential predictor variables of HAB for both SMR and GP methods; the selection criteria of the input variables after the initial set of variables is narrowed down; and an overview of the forecasting models used in this study.

3.1 Data Analysis Methods and Input Selection

The current study looked at structural models for forecasting HABs in Lake Erie. Both SMR and GP models need predictor variables as the inputs among the pre-selected variables presented in Chapter II (Tables 3 and 4). To determine the final important and effective variables, in the Chapter II the influence and importance on HABs for the variables was explored. In Lake Erie, the two key sources of nutrients for the HABs are nitrogen and phosphorous. Phosphorous considered as the critical nutrient which is required for metabolic reactions in plant life and in Lake Erie the limiting nutrient factor is nitrogen. Maumee River is the most significant river in the Maumee River watershed conveying all the nutrient to Western Lake Erie and thus the river discharge of Maumee

River directly related to amount of nutrient in Western Lake Erie. HABs are dependent on temperature also, and that's why the algae bloom starts in June and July and there is no bloom in January and February. Western Lake Erie has the shallowest part of Lake Erie, thus air temperature effects the water temperature. Another important factor on HABs is wind speed and direction, which not only affects the intensity and mass of the HABs, but also can control where the HAB travels. Wind speed is not a resonator always, because the high wind speed can disrupt bloom growth.

3.1.1 Bloom Loading Periods and Individual Correlations

First step is to find the best match for predictor variables among predictand variables. The study looked at chlorine (Chl-a) and found it not to be a good match, but CI had a good correlation with Q and nutrient such as phosphorous and nitrogen. All variables are calculated based on monthly average method, except total phosphorus which is total mass of each month, in this case bloom loading period is selected from March to June and average of this period is calculated and presents as a number which stands for each year from 2002 to 2011.

The Figure 4 presents the correlation between the representative predictor variables (Q, TP, and PM) against Chl-a, which all predictor variables are averaged for the loading period of March to June for each year from 2002 to 2011, and the Chl-a is the peak concentration of each month from 2002 to 2011.

Figure 4 indicates that there was an attempt to find the best correlation between all three variables against the peak Chl-a values. All the three coefficient of determination (R^2) were lower than 0.2 which is not statistically significant.

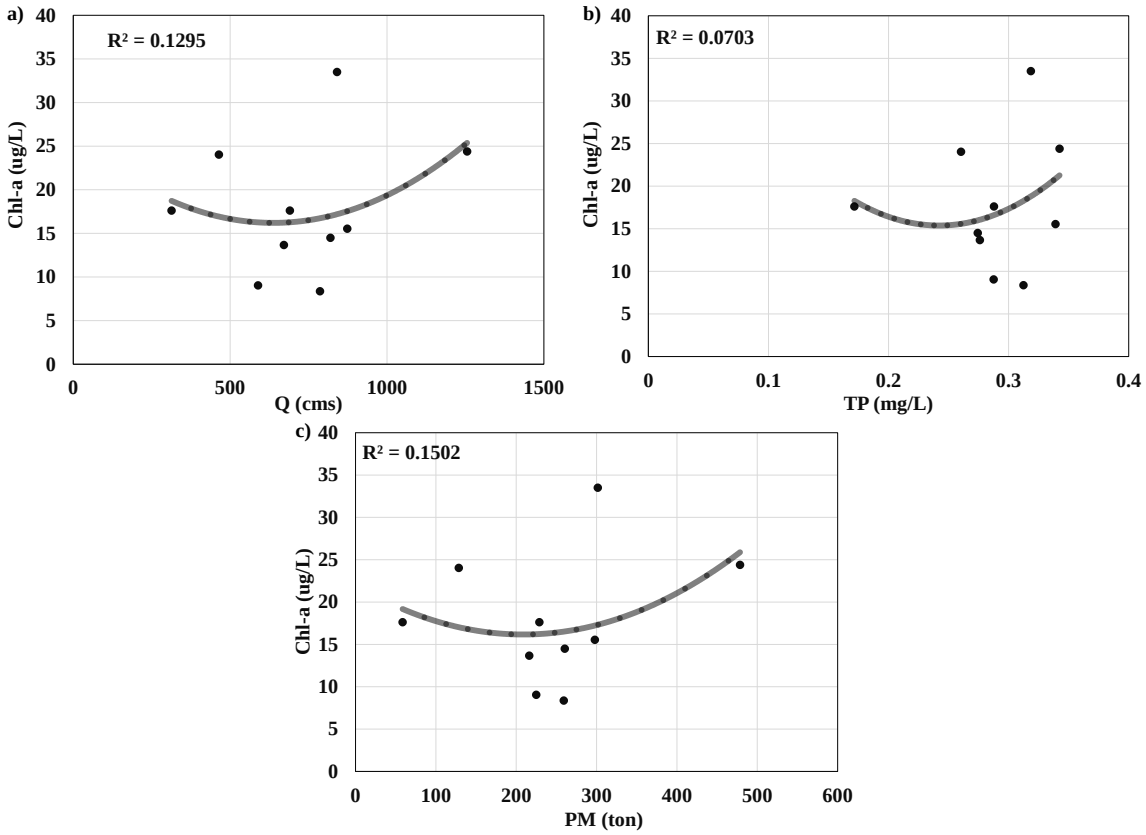


Figure 4. Correlation of Peak Chl-a and Nutrient Contributing Variables Averaged from March to June (a) Q vs. Peak Chl-a, (b) TP vs. Peak Chl-a, (c) PM vs. Peak Chl-a

The next step is to evaluate the correlation of same variables against CI instead of Chl-a. Figure 5 presents the correlation between those three predictor variables (Q, TP, and PM) against CI, which all predictor variables are averaged for the loading period (March to June) from 2002 to 2011, and the CI is the peak value of each month for from 2002 to 2011.

As presented in Figure 5(a), Q and CI shows the highest correlation, and it explains that nutrient transport heavily relies on rainfall-runoff. The correlation between TP and CI is weaker than Q vs. CI, yet still statistically significant. This study thus included TP as a predictor variable. The high correlation of PM and CI shows that there can be periods of high concentration of TP but lower precipitation resulting in a lower amount of

total phosphorus load, which is PM, entering the Lake. This study selected CI as a final response variable as an output for both SMR and GP models because CI showed higher correlation with predictor variables than Chl-a.

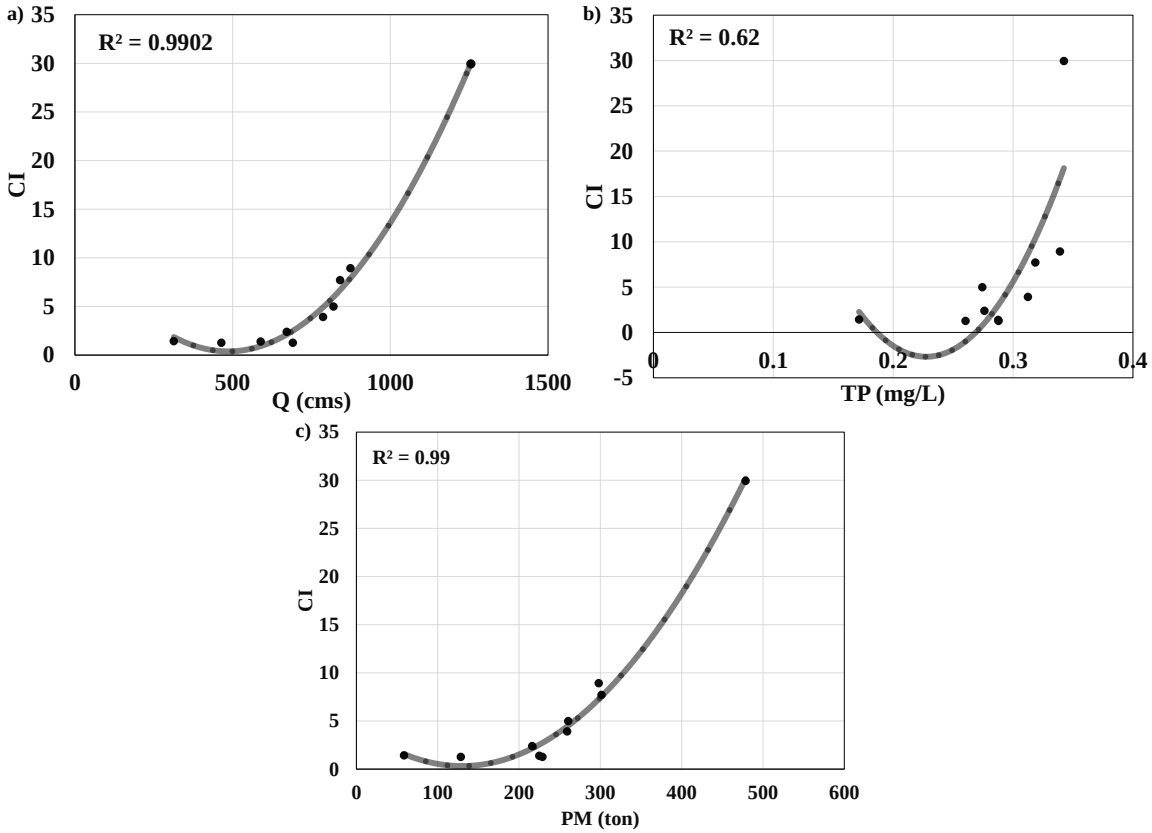


Figure 5. Correlation of Peak CI and Nutrient Contributing Variables Averaged from March to June (a) Q vs. Peak CI (b) TP vs. Peak CI (c) PM vs. Peak CI

Although phosphorus and nitrogen are the two main sources of nutrients for the HABs in Lake Erie, it is well known that phosphorus promotes bloom growth more, which is often the nutrient that there is less of in freshwater whereas nitrogen is the limiting nutrient factor in saltwater. The variables were analyzed and narrowed down to the eight predictors and one predictand variables based on the availability in the study area and correlation with CI (Table 3 and Figures 4 and 5) as presented in Table 5 and Table 6.

Table 5. List of Final Predictor Variables

Variable (Unit)		Source	Method
Q	Flow Rate (cfs)	NCWQR	Monthly average
TP	Total Phosphorus (mg/L)	NCWQR	Monthly average
PM	Phosphorus Mass (ton)	NCWQR	Monthly total
SRP	Soluble Reactive Phosphorus (mg/L)	NCWQR	Monthly average
TKN	Total Kjeldahl Nitrogen (mg/L)	NCWQR	Monthly average
Water	Water Temperature (°C)	USGS	Monthly average
Air	Air Temperature (°C)	USGS	Monthly average
Wind	Wind Speed (knots)	USGS	Monthly average

Table 6. Final Predictand Variable

Variable (Unit)		Source	
CI	Cyanobacteria Index (10^{20} cells)	NOAA	Monthly average

3.1.2 Spearman Rank Correlation Analysis

After selecting variables, various lengths of lag time and average period were used to compute predictor variables as an input to SMR and GP. For example, there is a delay in timing between Q and CI in Western Lake Erie (Figure 6). Many previous HAB studies revealed that nutrition features (i.e., both amount and timing) and seasonal water temperature are the most important watershed variables among many others that can substantially impact on HAB in a receiving waterbody. HAB is active and starts growing when the water temperature is over 25 °C (Indiana University, 2017). As the average water temperature of Western Lake is generally over 25 °C from July to October, another controlling factor should be the characteristics of nutrient loading. As discussed earlier, nutrition (Phosphorous and Nitrogen) is transported from the Maumee River basin during February to April or May and thus active growing of algal bloom bacteria is delayed (promoted) about 2 to 5 months until water temperature is amicable for HAB (Stumpf, 2012). However, timing of nutrition application and rainfall-runoff in the watershed are

not in agreement and varies year by year resulting not clear trend in lag time for each HAB month.

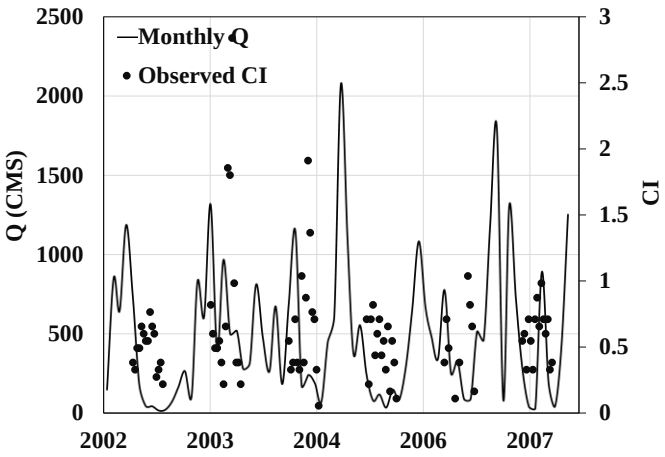


Figure 6. Observed CI and Monthly Q from 2002~2007

To consider this correlation, the Spearman rank correlation coefficient was used to determine the time to lag each variable was by analyzing the correlations between input variables and CI values.

First, various lag time and average period for all variables counted are presented in Table 7 for September as a demonstration to define the notations. Same methods were applied to July, August, and October.

Significant lag times and average periods were selected using the Spearman correlation coefficient that showed *p*-value less than 0.05. The selected variables are used as final inputs to both SMR and GP models. Through the analysis of individual correlations, the variables averaged over common time periods were eliminated to avoid multicollinearity.

Table 7. Various Lag Times and Average Periods for September

Lag	No. of months used in average	Months used in calculation	Lag and Period	Subscript for variables
1	1	Aug	t-1	1,1
2	1	Jul	t-2	2,1
3	1	Jun	t-3	3,1
4	1	May	t-4	4,1
5	1	Apr	t-5	5,1
6	1	Mar	t-6	6,1
1	2	Aug-Jul	t-1,t-2	1,2
2	2	Jul-Jun	t-2,t-3	2,2
3	2	Jun-May	t-3,t-4	3,2
4	2	May-Apr	t-4,t-5	4,2
5	2	Apr-Mar	t-5,t-6	5,2
1	3	Aug-Jul-Jun	t-1,t-2,t-3	1,3
2	3	Jul-Jun-May	t-2,t-3,t-4	2,3
3	3	Jun-May-Apr	t-3,t-4,t-5	3,3
4	3	May-Apr-March	t-4,t-5,t-6	4,3
1	4	Aug-Jul-Jun-May	t-1,t-2,t-3,t-4	1,4
2	4	Jul-Jun-May-Apr	t-2,t-3,t-4,t-5	2,4
3	4	Jun-May-Apr-March	t-3,t-4,t-5,t-6	3,4
1	5	Aug-Jul-Jun-May-Apr	t-1,t-2,t-3,t-4,t-5	1,5
2	5	Jul-Jun-May-Apr-March	t-2,t-3,t-4,t-5,t-6	2,5
1	6	Aug-Jul-Jun-May-Apr-March	t-1,t-2,t-3,t-4,t-5,t-6	1,6

Spearman method calculates ρ presented in Eq. (1) and then transforms ρ into a *p*-value by using exact permutation distributions. After transforming, the predictor variables with *p*-values less than 0.05, can represent high importance frequently in statistical analyses.

$$\rho = 1 - \frac{6 \sum {d_i}^2}{n(n^2 - 1)} \quad (1)$$

where ρ is Spearman rank correlation coefficient, d_i is difference in ranks between corresponding x and y variables, and n is total number of values in the data set. Two training periods from 2002 to 2011 and 2002 to 2014 were considered for Spearman method. Spearman method considered three hundred and thirty-six different

combinations of averaging and lag periods and all *p*-values calculated for the predictor variables separately for each bloom month for two training periods of 2002-2011 and 2002-2014.

After applying Spearman selection method, the selected variables with significant overlapped period were removed in order to reduce bias in the both SMR and GP models. When two sets of variables share longer than two-third of their average period, the shorter variable set is removed from the final set of inputs. For instance, if PM from March to June was selected from Spearman method, it would be removed if PM from February to July was also selected by Spearman method. The result of this step reduced the total number of variables and presents the finalized inputs for both models for each training period of 2002 to 2011 and 2002 to 2014. The final selection of input variables from the Spearman rank correlation analysis are shown in Table 8.

Table 8. Final Selected Inputs for SMR and GP Models by Spearman for Both Training Periods

Month	Jul		Aug		Sep		Oct	
Training period	(02-11)	(02-14)	(02-11)	(02-14)	(02-11)	(02-14)	(02-11)	(02-14)
Q	$Q_{5,1}$* $Q_{4,2}$ $Q_{3,3}$ $Q_{1,6}$	$Q_{3,1}$ $Q_{4,1}$ $Q_{6,1}$ $Q_{2,2}$* $Q_{3,2}$ $Q_{1,5}$	$Q_{5,1}$ $Q_{4,2}$ $Q_{1,6}$	$Q_{5,1}$* $Q_{4,2}$* $Q_{1,6}$*	$Q_{3,1}$ $Q_{6,1}$ $Q_{3,3}$ $Q_{1,6}$*	$Q_{3,1}$* $Q_{6,1}$ $Q_{3,3}$ $Q_{1,6}$	$Q_{5,2}$ $Q_{1,6}$*	$Q_{5,2}$ $Q_{1,6}$
TP	$TP_{1,5}$	$TP_{3,1}$ $TP_{1,5}$			$TP_{3,4}$*		$TP_{1,6}$*	$TP_{1,6}$
PM	$PM_{3,1}$ $PM_{4,1}$ $PM_{6,1}$ $PM_{2,2}$ $PM_{1,5}$*	$PM_{3,1}$* $PM_{2,2}$ $PM_{3,2}$ $PM_{1,5}$	$PM_{5,1}$ $PM_{1,6}$*	$PM_{5,1}$* $PM_{4,2}$* $PM_{1,6}$	$PM_{3,1}$ $PM_{3,3}$ $PM_{1,6}$*	$PM_{3,3}$* $PM_{1,6}$	$PM_{5,2}$ $PM_{1,6}$	$PM_{5,2}$* $PM_{1,6}$*
SRP			$SRP_{6,1}$ $SRP_{5,2}$					
TKN	$TKN_{5,2}$ $TKN_{3,3}$ $TKN_{4,3}$ $TKN_{1,6}$*	$TKN_{5,2}$ $TKN_{2,3}$ $TKN_{1,6}$*	$TKN_{1,6}$*		$TKN_{1,1}$* $TKN_{3,4}$*	$TKN_{3,4}$*	$TKN_{1,2}$ $TKN_{4,2}$	
Water	$Water_{3,4}$	$Water_{3,4}$	$Water_{2,1}$ $Water_{1,2}$ $Water_{2,2}$ $Water_{1,3}$* $Water_{1,6}$*	$Water_{2,1}$				
Air							$Air_{3,2}$ $Air_{3,4}$	$Air_{3,4}$
Wind							$Wind_{3,2}$ $Wind_{1,4}$*	$Wind_{3,2}$* $Wind_{1,4}$*

Note) Italicized and * marked variables were selected by SMR and GP, respectively.

3.2 Stepwise Multiple Regression Model

In the first prediction model, SMR was used to create a linear model that relates the predictor variables of the system linearly to a single predictand variable using Eq. (2) (Kalogirou & Sencan, 2010).

$$CI = \beta_0 + (\beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n) + (\beta_{n+1} X_1 X_2 + \cdots + \beta_m X_{n-1} X_n) + (\beta_{m+1} X_1 X_2 X_3 + \cdots) + \cdots + \beta_k X_1 X_2 \ldots X_n \quad (2)$$

where CI is the predictand variable, X_i represents a predictor variable, and β_i is the counterweight value of the predictor variables. A coefficient β_i measures the effect of each predictor variable taking into account the effect of all predictor variables in the model and it is calculated by the least squared error method. The regression coefficient for the *i* th predictor variable is the expected change in the predictand variable per unit change in the *i* th variable provided that all the other predictor variables are kept constant (Ramsami & Oree, 2015).

SMR is a modification of the forward collection so that after each phase in which a variable was selected, all considered variables in the model are analyzed to see if their significance has been decreased below the specified tolerance level. If a non-significant variable recognized by the model, SMR removes the founded variable from the model. SMR requires two significance levels: one for selecting variables and one for deselecting variables. To avoid an infinite loop of adding and removing in the procedure, the cutoff probability for deselecting variables should be greater than the cutoff probability for selecting variables.

In this study, the selection technique starts with unfilled set of predictors in the model. In each phase, it adds the predictor variable with the lowest *p*-value until there is no variables having *p*-value less than 0.05, the entrance tolerance. Then the exclusion technique is activated which eliminates from the model the variables with the largest *p*-value, if it is greater than 0.1, the exit tolerance. The selection and removal procedures

are repeated consecutively until there is no variable for elimination. Those processes were performed using the function ‘stepwiselm’ built in Matlab.

3.3 Genetic Programming Model

An evolutionary or genetic algorithm applies the ideologies of development found in nature to the problem of finding an optimal answer to a Solver problem. In a genetic algorithm, the problem is encoded in a series of bit strings that are operated by the algorithm; in an evolutionary algorithm, the problem functions and decision variables are used directly. Most commercial Solver products are based on Genetic Algorithms.

Rather than working on bit strings GP works on analyze trees, which GA use them to in a symbolic form estimate the equation that best defines how the output, which is the predicted variable in this paper, relates to the input variables, which are the predictors. The procedure considers a primary population of randomly generated equations, derived from the random combination of the given random numbers, pre-selected functions, and also given predictor variables as inputs. The pre-selected functions include arithmetic operators such as: plus, minus, multiply, divide, and power, mathematical operators such as: sin, cos, exp, and log, and transferring functions. Although preselection enables the GP simulation fast, functions must be properly selected based on reasonable understanding of the process. The population of potential answers is then subjected to an evolutionary process, and a measure of how well they solve the problem, the fitness, of the advanced programs are calculated. Finally, from the initial population, individual generated formulas are selected based on the fit to the target variables for a next iteration.

User must choose a number of GP limits before applying the algorithm to generate the formulas, such as generations’ population number and size as well as crossover and

mutation probability. The generated formula that fit the outputs less well are rejected. This development procedure is repeated over successive generations and is driven towards finding symbolic expressions describing the outputs, which can be scientifically interpreted to derive knowledge about the process being modeled. This study used Discipulus 5.2 (Francone, 1998) to build optimal GP models for HAB months in Western Lake Erie.

The generated formula performance is internally evaluated using two methods to measure the error and correlation, the root-mean-square-error (RMSE) and the correlation coefficient (CC) as defined in Eqs. (3) and (4).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}[(X_m)_i - (X_s)_i]^2} \tag{3}$$

$$CC = \sum_{i=1}^{n} \frac{[(X_m)_i - (\overline{X_m})][(X_s)_i - (\overline{X_s})]}{\sqrt{\sum_{i=1}^{n}[(X_m)_i - (\overline{X_m})_i]^2}\sqrt{\sum_{i=1}^{n}[(X_m)_i - (\overline{X_s})_i]^2}} \tag{4}$$

where X is any variable that is being forecasted; the subscripts m and s represent the measured and simulated values; the average value of the associated variable is represented by 'bar' above the variable; and n is the total number of training records.

To select the best model among the all generated models by GP, the process approved in this study is described as follow:

Step 1. Identification of the maximum and minimum value of CI in the time series.

Step 2. Separation of the time series into two groups: (i) for training the GP model in a specified range of CI and (ii) validating the GP model outside this range i.e., for values of CI close to its low and high extremes.

Step 3. The GP is trained with those input vectors that does not contain intermediate values of CI to avoid overlap.

Step 4. The GP-evolved models for each experiment are validated separately for the values that are close to the low and high extremes. This is done to test how various models perform for CI values that are extrapolated outside of the training range of CI.

Step 5. From the models obtained in Step 4 above, the best models with almost equal error measures are selected. These are then analyzed to determine their meaningfulness in explaining the physical aspects of the process.

Step 6. The best model obtained from Step 5 above is subjected to sensitivity analysis to identify the significance of the input variables.

GP is carried out for multiple runs using different factors including mutation rate, crossover rate, number of generations, population size, etc., which are adjusted by trial and error and are presented in Table 9 (Sivapragasam, 2010).

Table 9. Parameter Values Used in GP Runs for Discipulus

GP Parameter	value
Population size	500
Maximum equation size	50
Crossover rate	0.96
Mutation rate	0.05
Elitism used	Yes

CHAPTER IV

RESULTS AND DISCUSSION

4.1 Stepwise Multiple Regression Model

SMR generated different formulas for individual months using the predictor variables selected by the Spearman test, which are *Q*, *TP*, *PM*, *SRP*, *TKN*, *Water, Air* and *Wind* for various lag time and average period. For each month two different training periods were used; rom 2002 to 2011 training period to predict HABs of 2012 to 2015 and from 2002 to 2014 training period to predict 2015 HABs. Figures 7 to 10 present the former and latter results.

A SMR CI prediction model for July was trained for 2002-2011 and presented in Table 10 - Eq. (5). Five variables were automatically selected, which are $Q_{5,1}$, $TKN_{5,2}$, $TKN_{3,3}$, $TKN_{4,3}$, and $TKN_{1,6}$. The variables with higher importance than other variables in CI prediction for July are discharge (*Q*) and nitrogen concentration (*TKN*) in Western Lake Erie. Based on the training period of 2002 to 2011 in SMR, *Q* in February and *TKN* inflowed during June to January are correlated with the July CI values significantly. Eq.

(6) also presents CI prediction model for July for extended training period of 2002-2014. Selected variables in Eq. (6) are $Q_{6,1}$, $TP_{1,5}$, $PM_{3,1}$, $PM_{2,2}$, $PM_{3,2}$, and $PM_{1,5}$. The variables with more effects on CI prediction for July are discharge (*Q*), phosphorous concentration (*TP*), and total phosphorous mass (*PM*) in Western Lake Erie. With longer training period, nitrogen concentration lost its effects on CI prediction; however, phosphorous shows more affectivity in Eq. (6) than Eq. (5). Extending the training period shows that, *Q* in January has more effect than February, and phosphorous inflowed during June to February is correlated with July CI values meaningfully; However, PM in April has the highest effect on CI predicted for July 2015.

Table 10. SMR Prediction Model for Two Different Training Periods

Target Month	Training Period	Equation	Eq. No.
July	2002~2011	$-47.002 + 0.0024921\ Q_{5,1} - 165.74\ TKN_{5,2} + 1.3122\ TKN_{3,3} + 159.28\ TKN_{4,3} + 46.463\ TKN_{1,6} + 6.4971\ TKN_{5,2}TKN_{3,3} + 102.32\ TKN_{5,2}TKN_{1,6} - 116.75\ TKN_{4,3}TKN_{1,6}$	(5)
	2002~2014	$16.92 - 0.00065632\ Q_{6,1} - 98.88\ TP_{1,5} - 0.062375\ PM_{3,1} + 0.041184\ PM_{2,2} - 0.028678\ PM_{3,2} + 0.065138\ PM_{1,5} + 0.22066\ TP_{1,5}PM_{3,1} + 0.016089\ TP_{1,5}PM_{3,2} - 4.7082 \times 10^{-5}\ PM_{3,1}PM_{3,2} + 0.00018153PM_{2,2}PM_{3,2} - 0.00028966PM_{2,2}PM_{1,5}$	(6)
Aug.	2002~2011	$-2.5522 - 0.009642\ Q_{4,2} + 0.024167\ Q_{1,6} + 0.020692\ PM_{5,1} - 0.057463\ PM_{1,6} + 4.3077 \times 10^{-5}\ Q_{4,2}PM_{1,6} - 3.06 \times 10^{-5}\ Q_{1,6}PM_{5,1}$	(7)
	2002~2014	$33.15 + 0.010711\ Q_{1,6} + 0.19691\ PM_{5,1} - 0.37351\ PM_{4,2} - 0.044604\ PM_{1,6} - 1.7113\ Water_{2,1} - 0.00012461\ PM_{5,1}PM_{1,6} - 0.0091984\ PM_{5,1}Water_{2,1} + 0.0002285\ PM_{4,2}PM_{1,6}$	(8)
Sep.	2002~2011	$2.7139 + 0.00033942\ Q_{6,1} - 0.015274\ Q_{3,3} + 0.013085\ Q_{1,6} + 4.9469\ TKN_{1,1} - 67.17\ TKN_{3,4} + 0.1471\ Q_{3,3}TKN_{1,1} + 0.12714\ Q_{3,3}TKN_{3,4} - 0.19429\ Q_{1,6}TKN_{1,1}$	(9)
	2002~2014	$-44.173 - 0.0030401Q_{3,1} - 0.4514Q_{1,6} - 0.24826PM_{3,3} + 1.8759PM_{1,6} + 31.146TKN_{3,4} + 0.31372Q_{1,6}TKN_{3,4} + 0.0001405PM_{3,3}PM_{1,6} + 0.1506PM_{3,3}TKN_{3,4} - 1.2922\ PM_{1,6}TKN_{3,4}$	(10)
Oct.	2002~2011	$74.965 + 0.0088Q_{5,2} - 0.46538TP_{1,6} + 0.17332PM_{5,2} - 5.2045PM_{1,6} - 13.25\ Wind_{3,2} + 0.073485\ TP_{1,6}Wind_{3,2} - 0.02632\ PM_{5,2}Wind_{3,2} + 0.92584PM_{1,6}Wind_{3,2}$	(11)
	2002~2014	$-313.53 - 0.31369\ PM_{1,6} + 1.3896\ Air_{3,4} + 44.044\ Wind_{3,2} + 63.344\ Wind_{1,4} + 0.12524\ PM_{1,6}Wind_{3,2} - 0.078379\ PM_{1,6}Wind_{1,4} - 9.7783\ Wind_{3,2}Wind_{1,4}$	(12)

Training and prediction results are shown in Figure 7. Overall, the training performance in Table 11, shows R^2 = 0.99 for both 2002-2011 and 2002-2014 training periods. Predicted CI values for 2012 to 2014 are very close to the observed CI values.

However, the 2015 CI value is much underestimated because such high CI values have not included in the training period of July. This issue may be considered as one of weakness of data-driven models trained for a short period that may not include both high and low limit of observations; however, July CI predicted for 2015 with extended training period is more accurate than July CI prediction with shorter training Period. Based on Table 11, accuracy of the final SMR model for July increased significantly from R^2 = 0.52 with shorter training period to R^2 = 0.98 with extended training period.

Table 11. R^2 Values of SMR Models Trained for Two Different Training Periods

Target Month	Training R^2		Whole Model R^2	
	Training Period		Training Period	
	2002~2011	2002~2014	2002~2011	2002~2014
July	0.99	0.99	0.52	0.98
Aug.	0.99	0.98	0.91	0.53
Sep.	0.99	0.98	0.79	0.74
Oct.	0.99	0.99	0.95	0.94

Figure 7(a), shows the over-fitting of the model, because R^2 in training period is 0.99, but R^2 for whole model is 0.52. Over-fitting may be considered as one of weakness of data-driven models trained for a short period. Table 12 presents the observed CI and estimated CI for 2015, which shows July CI 2015 with longer training period is more accurate than July CI 2015 with shorter training period. Based on Table 11, the SMR model for August does not present an accurate formula, regarding lower accuracy in short training period than extended training period.

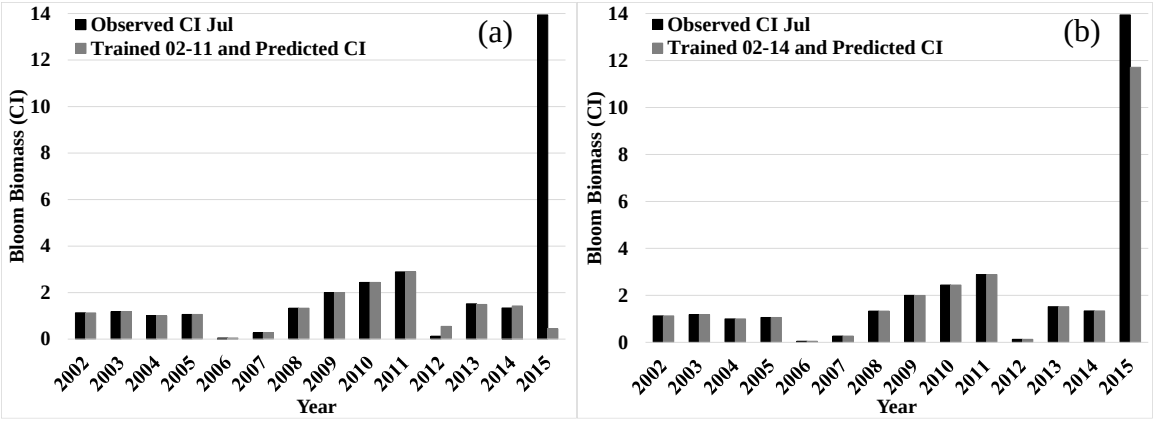


Figure 7. SMR Results for July Prediction with Training Period of (a) 2002~2011 and (b) 2002~2014

Table 12. 2015 CI Values Predicted by SMR Trained for Two Different Training Periods.

Target Month	CI_{2015}	Training Period	
		2002~2011	2002~2014
July	13.94	0.45	11.71
Aug.	29.2	18.60	4.56
Sep.	16.95	37.80	4.22
Oct.	7.07	9.32	12.30

A SMR CI prediction model for August was trained for 2002-2011 and presented in Table 10 - Eq. (7). Five variables were automatically selected, which are $Q_{4,2}$, $Q_{1,6}$, $PM_{5,1}$, and $PM_{1,6}$. The variables with higher significance than other variables in CI prediction for August are discharge (*Q*) and *PM* in Western Lake Erie. Based on the training period of 2002 to 2011 in SMR, *Q* in March and *PM* amount in April and March are correlated with the August CI values significantly; however, *Q* and *PM* in July to February have a meaningful correlation. Eq. (8) also displays CI prediction model for August for extended training period of 2002-2014. Selected variables in Eq. (8) are $Q_{6,1}$, $PM_{5,1}$, $PM_{4,2}$, $PM_{1,6}$, and $Water_{2,1}$. The variables with more effects on CI prediction for August are discharge (*Q*), *PM* like Eq. (7), and water temperature (*Water*) in Western Lake Erie. Longer training period shows that water physical characters will get involved

with CI prediction in long-term period. Extending the training period shows that, *Q* in February, phosphorous total mass amount in July to February, and finally water temperature in May and April is correlated with August CI values meaningfully; However, *PM* in March has the highest effect on CI predicted for August 2015.

Training and prediction results of August are shown in Figure 8. Overall, the training performance in Table 11, shows $R^2 = 0.99$ for 2002-2011 training period and $R^2 = 0.98$ for 2002-2014 training period, implying the training no longer improves under current training data set up to 2014. Predicted August CI values for 2012 and 2014 is very close to the observed CI values. However, the 2013 and 2015 August CI values are underestimated in training period 2002-2011 and 2002-2014, respectively. CI predicted for 2015 with extended training period does not show improvement compared to shorter training period (Figure 8(b) and Table 12). It is assumed that the HAB in August 2015 is a unique event that is hard to represent under given data set because R^2 in training period is 0.98 and it deteriorates to 0.53 for all period.

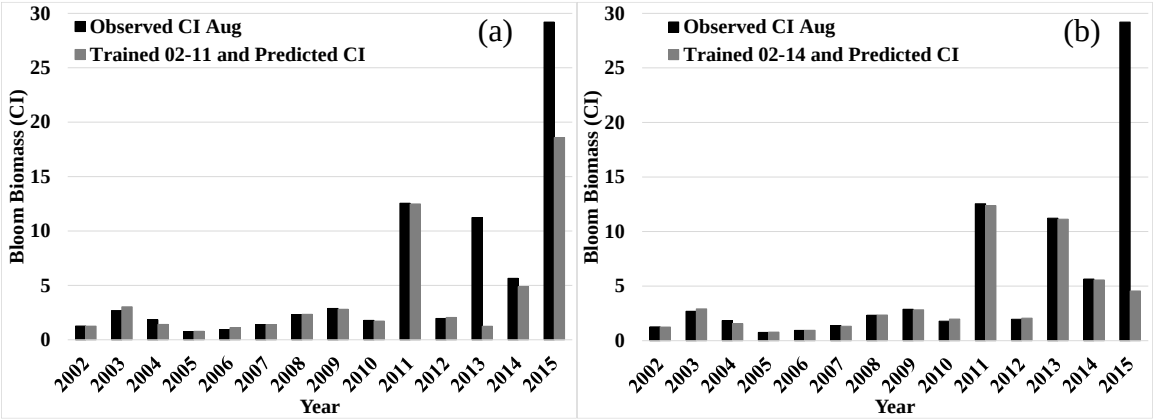


Figure 8. SMR Results for August Prediction with Training Period of (a) 2002~2011 and (b) 2002~2014

A SMR CI prediction model for September was trained for 2002-2011 and presented in Table 10 - Eq. (9). Five variables were automatically selected, which are $Q_{6,1}$, $Q_{3,3}$, $Q_{1,6}$, $TKN_{1,1}$, and $TKN_{3,4}$. The variables with higher significance than other variables in CI prediction for September are *Q* and *TKN* in Western Lake Erie. Based on the training period of 2002 to 2011 in SMR, *Q* and *TKN* inflowed in June to March is correlated with the September CI significantly; however, August *Q* and *TKN* is correlated to September CI meaningfully. Eq. (10) also displays CI prediction model for September for extended training period of 2002-2014. Selected variables in Eq. (10) are $Q_{3,1}$, $Q_{1,6}$, $PM_{3,3}$, $PM_{1,6}$, and $TKN_{3,4}$. The variables with more impacts on CI prediction for August are *Q*, *PM*, and *TKN*. Longer training period shows that phosphorous is another variable to predict September CI value. Extending the training period shows that, *Q* in June and *PM* from June to April are correlated with September CI values significantly; however, *Q* and *PM* in August and July and *TKN* in June to March has a meaningful correlation with September CI.

Training and prediction results of September are shown in Figure 9. Overall, the training performance in Table 11 shows $R^2 = 0.99$ for 2002-2011 training period and $R^2 = 0.98$ for 2002-2014. Similar to August, training does not improve significantly by adding three more training data (i.e., observations). In Figure 9(a), September CI in all years except 2013 shows over-estimation in prediction period. In Figure 9(b), September CI 2015 is much underestimated even with longer training period. This observation is consistent with August 2015, which is hard to detect the consistent mechanism of HAB occurred in 2015.

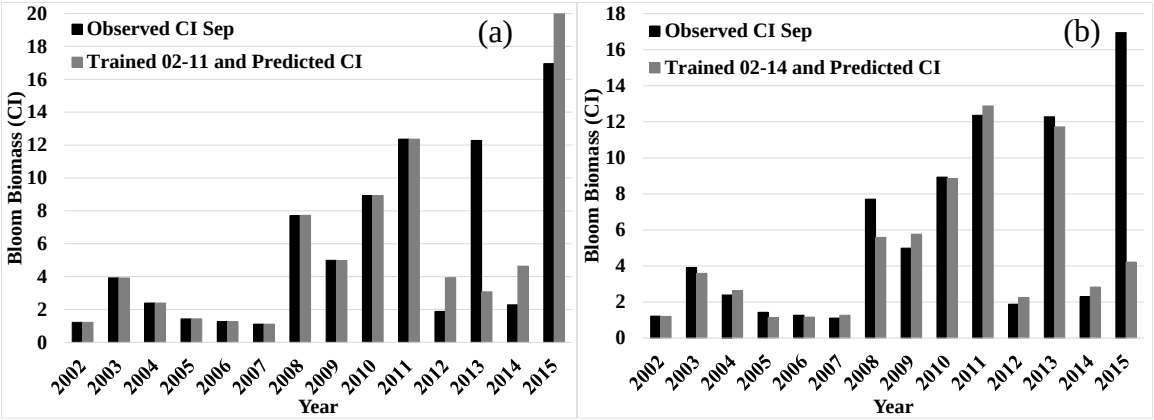


Figure 9. SMR Results for September Prediction with Training Period of (a) 2002~2011 and (b) 2002~2014

A SMR CI prediction model for October was trained for 2002-2011 and presented in Table 10 - Eq. (11). Five variables were automatically selected, which are $Q_{5,2}$, $TP_{1,6}$, $PM_{5,2}$, $PM_{1,6}$, and $Wind_{3,2}$. The variables with higher importance than other variables in CI prediction for October are *Q*, *TKN*, *TP*, *PM*, and *Wind*. Based on the training period of 2002 to 2011 in SMR, *Q* in April and May, TP and PM in September to April, and *Wind* speed in July and June are correlated with the October CI values meaningfully. Eq. (12) also presents CI prediction model for October for extended training period of 2002-2014. Selected variables in Eq. (12) are $PM_{1,6}$, $Air_{3,4}$, $Wind_{3,2}$, and $Wind_{1,4}$. With longer training period, physical characteristics of Lake Erie involved more variables such as wind speed and air temperature; however, discharge has indirect effect on October CI because PM includes discharge in its calculation. Extending the training period shows that, *PM* in September to April, air temperature of July to April, wind speed during September to June is correlated with October CI values significantly.

Training and prediction results of October are shown in Figure 10. Overall, the training performance in Table 11 shows $R^2 = 0.99$ for both 2002-2011 and 2002-2014 training period. SMR model in October is more accurate than previous months, the

accuracy for the whole period is $R^2 = 0.95$ for both training periods. However, October CI for 2013 is significantly underestimated in short training period, while October CI for 2012 is overestimated compared to longer training period. This inconsistent observation in particular months is still in question to analyze clearly when longer observations (e.g., more than 30 years) are available to train the model and will explain this issue better in the future. Interestingly, October CI 2015 is predicted in more acceptable and consistent accuracy in both training period than other months.

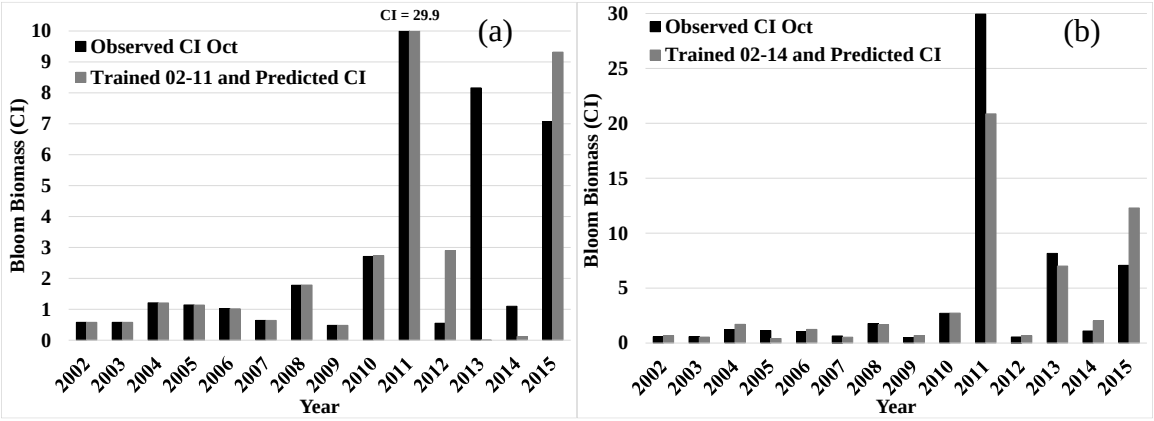


Figure 10. SMR Results for October Prediction with Training Period of (a) 2002~2011 and (b) 2002~2014

The individual SMR models trained for the two training periods are aggregated in Figure 11 to present the overall performanc of the SMR models. R^2 values for the training are 0.99 and 0.98 for the period of 2002-2011 and 2002-2014, respsectively. R^2 values for the whole prediciton model are 0.78 and 0.80, respectively.

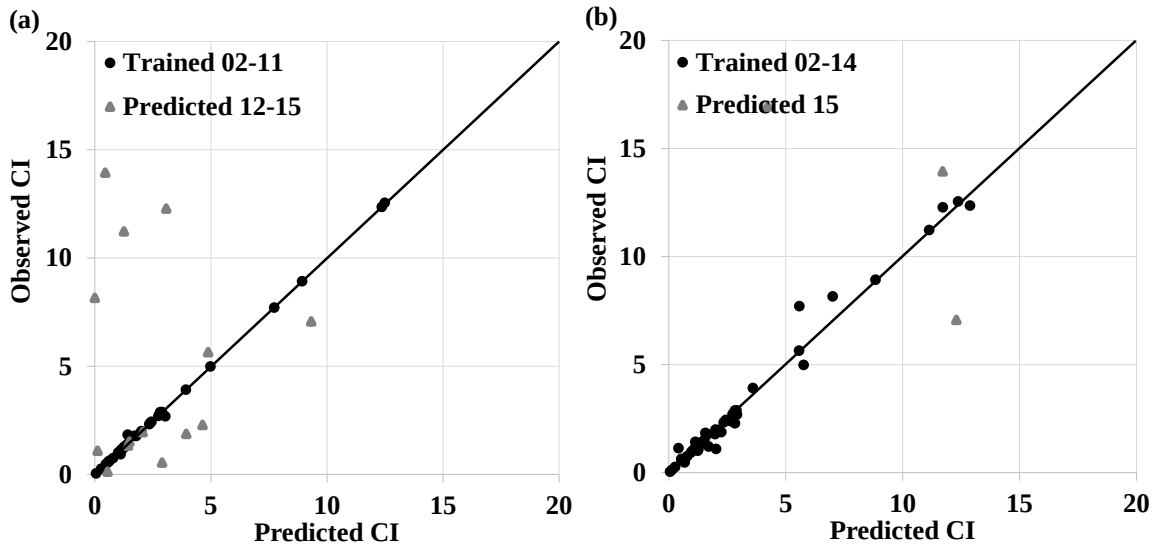


Figure 11. Comparison Between Observed and Predicted Results of SMR (a) Training Period of 2002~2011 and (b) Training Period of 2002~2014

The SMR model when trained up to 2011 (Figure 11(a)) is able to forecast the lower magnitude blooms well. However, the model had most of the predictions underpredicted for the higher magnitude blooms. When increasing the training period to 2014, the new SMR model is able to forecast the low magnitude blooms well and is able to forecast the higher magnitude blooms better when compared to the shorter training period.

Table 13 summarizes the selected variables by SMR models for each month. Commonly, discharge (*Q*) was selected in all HAB months except October with 2002-2014 training period. Then, nutrient and other climatic variables were selected as predictors by SMR automatically while air temperature and wind speed affect only October HAB events. This remains as future analysis when more solid evidences are available. It should be noted that the SMR model can be substantially improved when trained for a wide range of target values.

Table 13. SMR Variable Used for Both Training Periods

	July		August		September		October	
Variable	**(02-11)**	**(02-14)**	**(02-11)**	**(02-14)**	**(02-11)**	**(02-14)**	**(02-11)**	**(02-14)**
Q	×	×	×	×	×	×	×	
TP		×					×	
PM		×	×	×		×	×	×
SRP								
TKN	×				×	×		
Water				×				
Air								×
Wind							×	×

NOTE: Empty boxes represent variables that were not considered.

4.2 Genetic Programming Model

GP generates optimal prediction models (i.e., formulas) for individual months using all Spearman-selected inputs. Similar to SMR, for each month two different training period were used; from 2002 to 2011 and from 2002 to 2014. Prediction was made up to 2015. GP generated about 20 different formulas at each run for each month that show equal performance of training. Generated formulas are highly nonlinear and hard to explain in physical sense because GP finds “optimal numerical solution” disregarding physical mechanism of HABs. Table 14 presents the optimal formulas for HAB prediction generated by GP based on both short and long training periods.

Table 14. GP Prediction Model for Two Different Training Periods

Target Month	Training Period	Equation	Eq. No.
July	2002~2011	$1.7786 - \left\{\sin(\cos(f_2)) + \frac{f_1}{f_2}\right\}$ * $f_1 = -\frac{3.0994 \times TKN_{1,6}}{Q_{5,1}}$ * $f_2 = \cos\left(2 \times \left[\lvert f_1 \rvert + PM_{1,5} + 1.9876\right]\right)$	(13)
	2002~2014	$\left[\left\lvert \sin\left(2 \times PM_{3,1} - Q_{2,2}\right)\right\rvert \times 1.0842 + TKN_{1,6} - 1.3640 + f_3\right]^2$ * $f_1 = 2 \times \sin\left(\sin^2\left(\cos\left(0.9178 + PM_{3,1}\right) - 0.4427 + PM_{3,1}\right)\right)$ * $f_2 = \frac{0.9178 + PM_{3,1}}{f_1}$, * $f_3 = \frac{f_2}{\left(2 \times PM_{3,1}\right)^2}$	(14)
Aug.	2002~2011	$\frac{\lvert f_2 \rvert}{0.1241} \times \left(2^{TRUNC(f_3)}\right)^2 \times TKN_{1,6}$ * $f_1 = \cos\left(\cos\left(-1.5547 \times Water_{1,6}\right)\right)$ * $f_2 = \cos\left(\cos\left(\cos\left(-1.5547 \times Water_{1,6}\right)\right) + Water_{1,3} - PM_{1,6} + Water_{1,3}\right)$ * $f_3 = -f_1 - f_2$ * $TRUNC(f_3)$ means return the integer number of f_3	(15)
	2002~2014	$16 \times (f_2)^8 + 0.9178$ * $f_1 = \frac{1.1967 + \left(\frac{Q_{5,1} - PM_{5,1}}{0.7234 \times Q_{4,2}}\right)^2 + Q_{1,6}}{0.7234 \times Q_{4,2}}$ * $f_2 = \cos\left(\left[\frac{\left(\frac{\frac{16 \times (f_1)^{16}}{PM_{4,2}} + Q_{5,1} - PM_{5,1}}{0.7234 \times Q_{4,2}}\right)^8 + Q_{1,6}}{0.7234 \times Q_{4,2}}\right]^2 + 1.3648 - PM_{5,1}\right)$	(16)
Sep.	2002~2011	$\left\lvert f_3 - \frac{f_2}{f_3}\right\rvert \times TKN_{3,4}$ * $f_1 = \sin\left(TKN_{1,1} \times PM_{1,6} \times \left[\sin\left(\sin(2.2038) + 0.8215 - Q_{1,6} + TKN_{3,4}\right) + 4.1201\right] - 0.5916\right) + 0.7284 - Q_{1,6}$ * $f_2 = -f_1 \times \sin(f_1)$, * $f_3 = PM_{1,6} \times \left\{\frac{[\sin(f_1) + 2.2941] \times TKN_{1,1}}{TP_{3,4}} - 0.6762\right\}$	(17)
	2002~2014	$\sqrt{1.0868 \times \left[1.2592 + (2 \times f_1)^2 \times PM_{3,3} \times 0.0328\right] \times \left(TKN_{3,4}\right)^3}$ * $f_1 = \left[\sin\left(Q_{3,1} - 0.55\right)\right]^8$	(18)
Oct.	2002~2011	$2 \times TP_{1,6} + \left\lvert \sin\left(-\frac{[\sin(f_1) \times f_2]^2 + 1.7448}{f_2} + Q_{1,6}\right) \times f_2\right\rvert$ * $f_1 = -2 \times \cos\left(wind_{1,4}\right) + 0.1401$, * $f_2 = \frac{1}{\sin(f_1)}$	(19)
	2002~2014	$\cos\left(\frac{f_2 \times 2^{TRUNC(f_3)}}{PM_{5,2}}\right) \times 2^{TRUNC(f_3)}$ * $f_1 = -\left\lvert 4 \times \cos\left(-0.9765 \times \left\lvert \sin\left(4 \times \cos\left(1.5103 \times PM_{5,2}\right)\right) - PM_{1,6} - wind_{3,2}\right\rvert\right) \times 1.7786$ * $f_2 = -\frac{f_1}{PM_{5,2}} - wind_{1,4}$, * $f_3 = f_1 - f_2$ * $TRUNC(f_3)$ means return the integer number of f_3	(20)

The July CI prediction model trained for 2002-2011 is expressed in Table 14 - Eq. (13) showing $Q_{5,1}$, $TKN_{1,6}$, and $PM_{1,5}$ were selected as predictors. These variables are *Q* of February, *TKN* averaged from June to January, and *PM* averaged from June to February in Western Lake Erie. Compared to the SMR July model, $Q_{5,1}$ is commonly selected and *TKN* during April to January was commonly included. Eq. (14) also presents CI prediction model for July with extended training period of 2002-2014. Selected

variables in Eq. (14) are $Q_{2,2}$, $TKN_{1,6}$, and $PM_{3,1}$. The variables with more effects on CI prediction for July are *Q*, *PM*, and *TKN*. With longer training period, *TKN* of same lag time and average period is selected by GP with different lag time and average period of *Q* and *PM*. Commonly, *Q*, *PM*, and *TKN* are the key factors for July HAB prediction in both training periods. Extending the training period shows that, *PM* in April, *Q* of May and April, and *TKN* inflowed during June to January are correlated with July CI values significantly.

Training and prediction results of July are shown in Figure 12. In Table 15, R^2 for training period is 0.60 and predicted CI values for 2012 to 2014 are very close to the observed, except 2015, similar to the July SMR model. As GP could not train the model using 2002-2012 target values for the very high CI value observed in 2015, it is underestimated but superior than SMR result. This issue may be considered as one of weakness point of machine learning modeling, which cannot find an exact relationship between inputs and outputs while the output is unpredictably out of the regular range. R^2 for all period is 0.98 showing reasonable prediction performance. Extending the training period increases the R^2 from 0.60 to 0.64 (Table 6).

Table 15. R^2 Values of GP Models Trained for Two Different Training Periods

Target Month	Training R^2		Whole Model R^2	
	Training Period		Training Period	
	2002~2011	2002~2014	2002~2011	2002~2014
July	0.596	0.640	0.981	0.983
Aug.	0.993	0.982	0.412	0.923
Sep.	0.983	0.959	0.915	0.973
Oct.	0.998	0.996	0.986	0.996

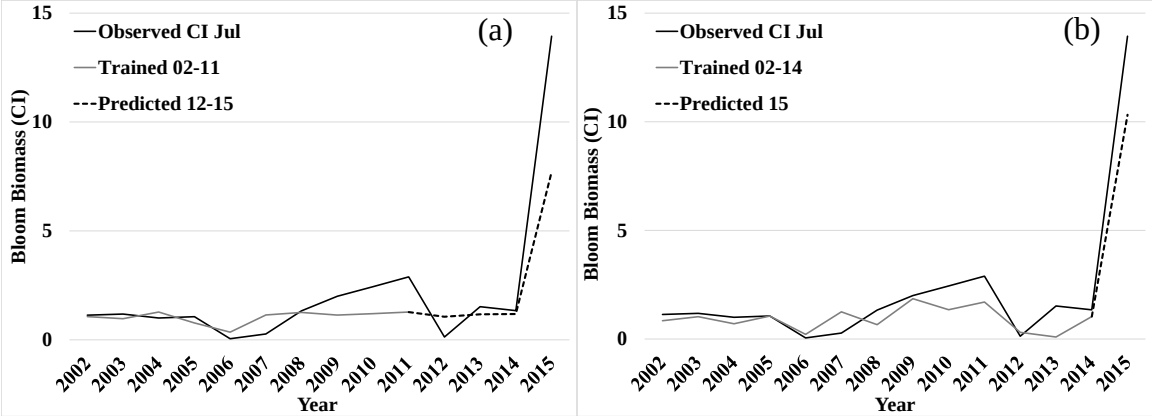


Figure 12. GP Results for July Prediction with Training Period of (a) 2002~2011 and (b) 2002~2014

As presented in Table 15 clearly, accuracy of the model for both training part and whole prediction generally increases with extending the training period. R^2 is increased from 0.60 to 0.98 in training period, and the whole model accuracy also is increased from 0.64 to 0.98. The results in Table 16, shows that July CI predicted for 2015 with longer training period is closer to observed July CI than July CI predicted with short training period. The final R^2 values of both periods are similar indicating the training is consistent and not overfitted.

Table 16. 2015 CI Values Predicted by GP Trained for Two Different Training Periods

Target Month	CI_{2015}	Training Period	
		2002~2011	2002~2014
July	13.94	7.80	10.32
Aug.	29.20	2.70	16.86
Sep.	16.95	12.61	15.18
Oct.	7.07	3.21	7.42

The August CI prediction model trained for 2002-2011 is expressed in Table 14 - Eq. (15) showing $Water_{1,6}$, $Water_{1,3}$, $PM_{1,6}$, and $TKN_{1,6}$ were selected as predictors. These variables are *PM* of July to February, *TKN* averaged from July to February, and *Water*

averaged from July to February in Western Lake Erie. It is interesting that *Water* shows a higher correlation with August CI than other months. Compared to the SMR August model, $PM_{1,6}$ is commonly selected and *TKN* and *Water* never selected by SMR which presents water physical characteristics has more effect on GP modeling than SMR. Eq. (16) also presents CI prediction model for July with extended training period of 2002-2014. Selected variables in Eq. (16) are $Q_{5,1}$, $Q_{4,2}$, $Q_{1,6}$, $PM_{5,1}$and $PM_{4,2}$. The variables with more effects on CI prediction for August are *Q* and *PM*. With longer training period phosphorous and discharge became the two key factors of GP model for August. Extending the training period shows that, *PM* in March and April, and discharge rate during July to February are correlated with August CI values meaningfully; however, *PM* and *Q* in March are significant.

Training and prediction results of August are shown in Figure 13. In Table 15, R^2 for training period is 0.99 and predicted CI values for 2012 to 2014 are very close to the observed except 2015, similar to the August SMR model that underestimates 2015. R^2 for whole August GP model is 0.41, which is very lower than 0.99 showing an over-fitting. This issue may be considered as one of weakness point of machine learning modeling that tends to find overfitted relationship between inputs and outputs when training period is short. However, extending the training period increases the R^2 value 0.41 to 0.92 for the whole period dramatically. For August 2015 prediction (Table 16), longer training period predict better than shorter training period, which can be improved more with more observed data in the future.

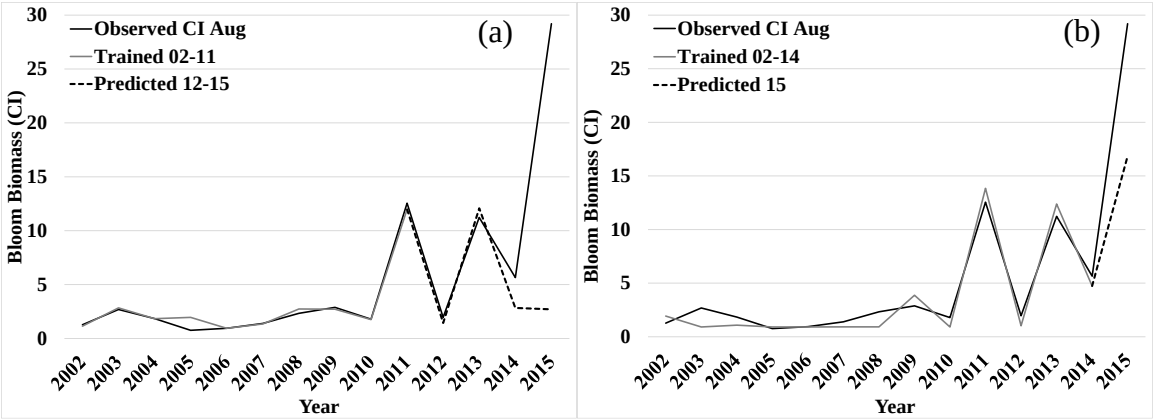


Figure 13. GP Results for August Prediction with Training Period of (a) 2002~2011 and (b) 2002~2014

The September CI prediction model trained for 2002-2011 is expressed in Table 14 - Eq. (17) showing $Q_{1,6}$, $TP_{3,4}$, $PM_{1,6}$, $TKN_{1,1}$, and $TKN_{3,4}$ were selected as predictors. These variables are *Q*, *TP* and *PM* of August to March, and *TKN* averaged from June to March and August. It is notable that *TP* averaged of June to March shows a higher correlation with September CI than other months. Compared to the SMR September model, $Q_{1,6}$, $TKN_{1,1}$, $TKN_{3,4}$ are commonly selected; however, both GP and SMR do not select any water-related physical characteristics. Eq. (18) also presents the CI prediction model for September with extended training period of 2002-2014. Selected variables in Eq. (18) are $Q_{3,1}$, $PM_{3,3}$, and $TKN_{3,4}$. For September CI prediction, GP commonly selected *Q*, *PM*, and *TKN*. With longer training period $TKN_{3,4}$ selected again, which illustrates that nitrogen is one of key factors of the September HAB model trained by GP.

Training and prediction results of September are shown in Figure 14. In Table 15, R^2 for training period of 2002-2011 is 0.98 and predicted CI values for 2012 to 2015 are very close to the observed in which SMR was not able to predict such accurately. R^2 for whole September GP model is 0.91 which shows there is no over fitting issue in this model again. Extending the training period increases the R^2 value from 0.91 to 0.97 for

whole model. The results in Table 16 shows that September CI predicted for 2015 with longer training period is closer to observed CI than the predicted CI by shorter training period.

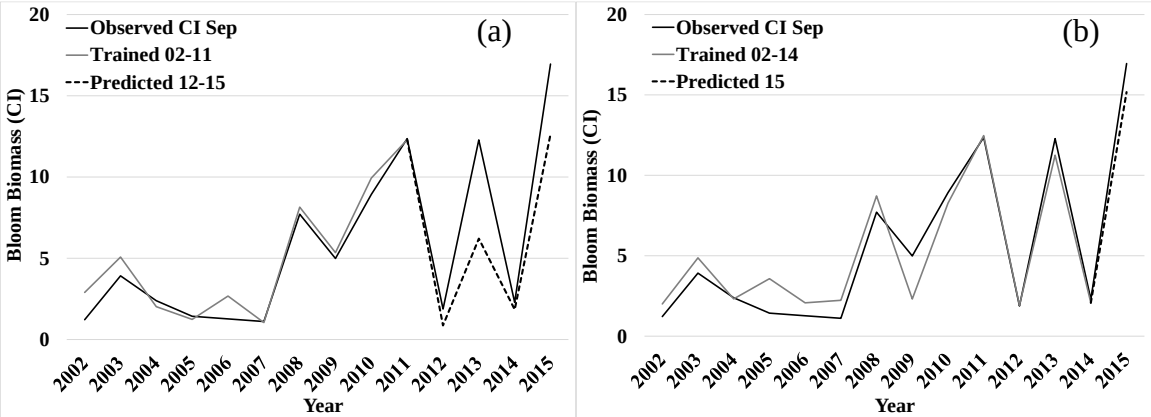


Figure 14. GP Results for September Prediction with Training Period of (a) 2002~2011 and (b) 2002~2014

The October CI prediction model trained for 2002-2011 is expressed in Table 14 - Eq. (19) showing $Q_{1,6}$, $TP_{1,6}$, and $wind_{1,4}$ were selected as predictors. These variables are *Q* and *TP* of September to April, and *wind* averaged from June to September. Compared to the SMR of October, $TP_{1,6}$ is commonly selected and *wind* in July and June is also selected. Eq. (20) also presents CI prediction model for October with extended training period of 2002-2014. Selected variables in Eq. (20) are $PM_{1,6}$, $PM_{5,2}$, $wind_{1,4}$, and $wind_{3,2}$. It is noted that *PM* is selected instead of Q and TP. It is logical that the product of Q and TP results in PM mathematically, implying that GP finds an equiprobable set of predictor variables. Extending the training period shows that *PM* in September to April, and wind speed in September to June are correlated with October CI values significantly along with extra variables *PM* averaged for May to April and wind speed averaged for July and June.

Training and prediction results of October are shown in Figure 15. In Table 15, R^2 for training period of 2002-2011 is 0.99 and predicted CI values for 2012 to 2015 are very close to the observed, in which SMR overestimated 2015. R^2 for the whole period is 0.98 in shorter training period and improves to 0.99 when trained for longer training period (Table 15).

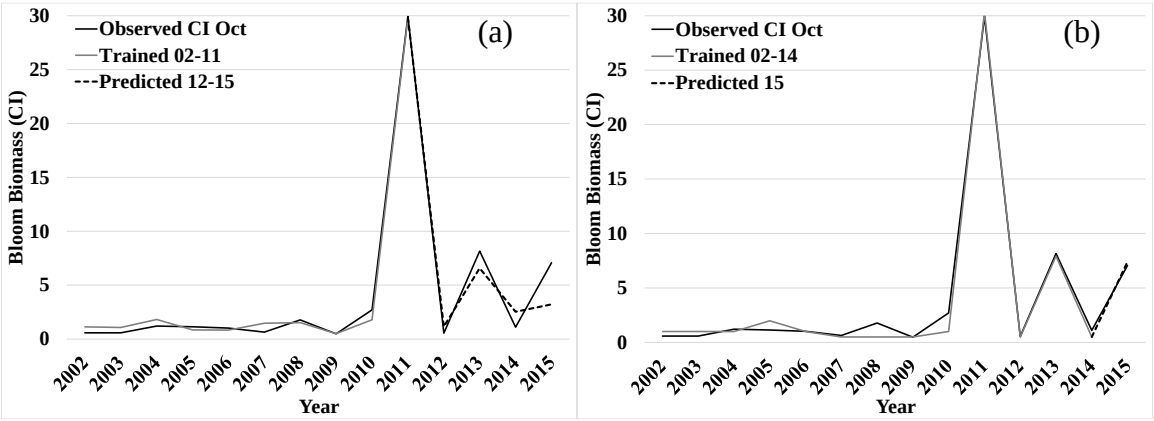


Figure 15. GP Results for October Prediction with Training Period of (a) 2002~2011 and (b) 2002~2014

The individual GP models trained for the two training periods are aggregated in Figure 16 to present the overall performanc of the SMR models. R^2 values for the training are 0.98 and 0.99 for the period of 2002-2011 and 2002-2014, respsectively. R^2 values for the whole prediciton period are 0.80 and 0.96, respectively.

The GP model when trained up to 2014 is able to forecast the blooms very well except July 2015 that exceed the range of the training data. This untrained data has been better predicted by GP than SMR because GP is capable of detecting highly nonlinear process hidden in data.

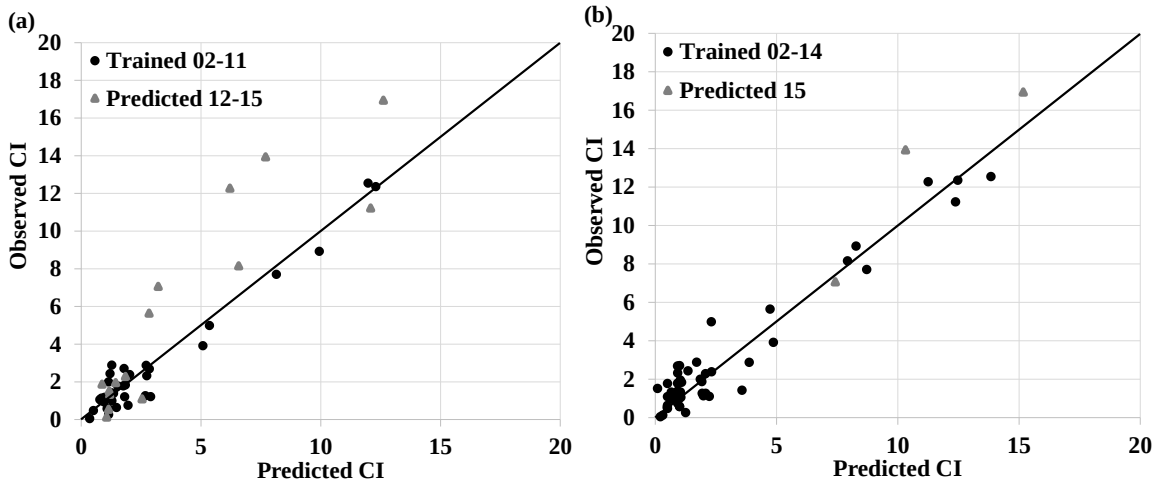


Figure 16. Comparison Between Observed and Predicted Results of GP with (a) Training Period of 2002~2011 and (b) Training Period of 2002~2014

Table 17 summarizes the selected variables by GP models for each month. Commonly, *Q* was selected in all HAB months except August with 2002-2011 training period and October with 2002-2014 training period. Then, nutrient and other climatic variables were selected as predictors by GP automatically while wind speed affect only October and water temperature affect only August HAB events. This remains as future analysis when more solid evidences are available. It should be noted that the GP model can be substantially improved when trained for a wide range of target values.

Table 17. GP Variable Selected for Both Training Periods

	July		August		September		October	
Variable	**(02-11)**	**(02-14)**	**(02-11)**	**(02-14)**	**(02-11)**	**(02-14)**	**(02-11)**	**(02-14)**
Q	×	×		×	×	×	×	
TP					×		×	
PM	×	×	×	×	×	×		×
SRP								
TKN	×	×	×		×	×		
Water			×					
Air								
Wind							×	×

NOTE: Empty boxes represent variables that were not considered.

4.3 Comparison of SMR and GP models

As discussed in Chapters 4.1 and 4.2, GP showed superior performance than SMR. SMR is a process to find an optimal linear model to explain target variables by considering both individual and product of predictor variables, while GP searches numerous numerical expressions to represent the nonlinear relationship between predictors and predictand. SMR is relative easy to express its formula compared to GP. Both models showed weakness in predicting target values that were not included in the range of training data. However, GP predicted substantially better than SMR in September, both training period, and October for long training period in 2015, although 2015 HAB events were not trained by GP. As SMR is a multi-variables linear model, there is a limitation to represent highly nonlinear behaviors that GP can detect and predict more easily. Very complicated and inexplicable mathematical expressions generated by GP are one of its drawbacks.

Model performance of July by two models is compared Figure 17(a). Although both models overpredict in 2012, 2013 and 2014 are well predicted by both models. GP shows better performance in 2015 than SMR and predicted the CI as “significant (CI above 7)” same to observation although the predicted values is 45% less than the observed, while SMR predicted July CI 2015 as “safe (CI less than 2)”. The most important reason is the lack of various data in the training period in which all CI observations are less than 3.

In Figure 17(b), both models are well predicted with extended training period. Both models predicted July CI 2015, as “significant” same to observation. July CI predicted by SMR and GP is almost 25% less than July CI observation; however, the model did not train for any significant CI. SMR shows better fit in training period than GP, which may

be concluded that, with increasing the training period SMR may be more accurate than GP, although SMR is a basic method in comparison with GP.

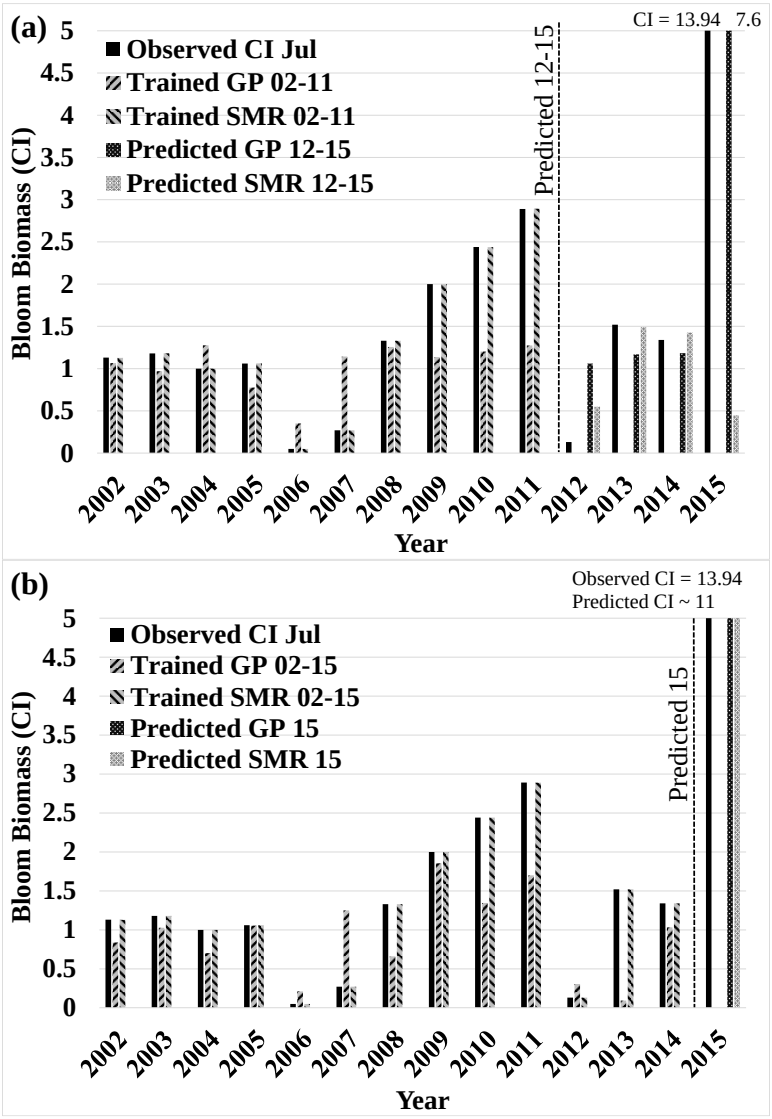


Figure 17. GP and SMR July Results: (a) 2002~2011 Training Period and (b) 2002~2014 Training Period

Model performance of August by two models is compared Figure 18(a) for training period 2002-2011. Both models for 2012 shows a good performance, because the range of August CI 2013 is similar to the range of training period. Although SMR prediction is underestimate and GP shows a better result in 2013, in 2014 and 2015 SMR performed

better than GP and all GP prediction was underestimated in comparison with observed August CI. SMR shows better performance in 2015 than GP and predicted the CI as "significant" same to observation although the predicted values is 35% less than the observed, while GP predicted August CI 2015 as "mild".

In Figure 18(b), both models are not well predicted with extended training period. GP August CI prediction is almost 40% less than August CI observation; however, both observed and predicted CIs are in the "significant" zone. In the other hand, SMR August CI prediction is in the "mild" zone and extending the training period did not improve the SMR results in 2015.

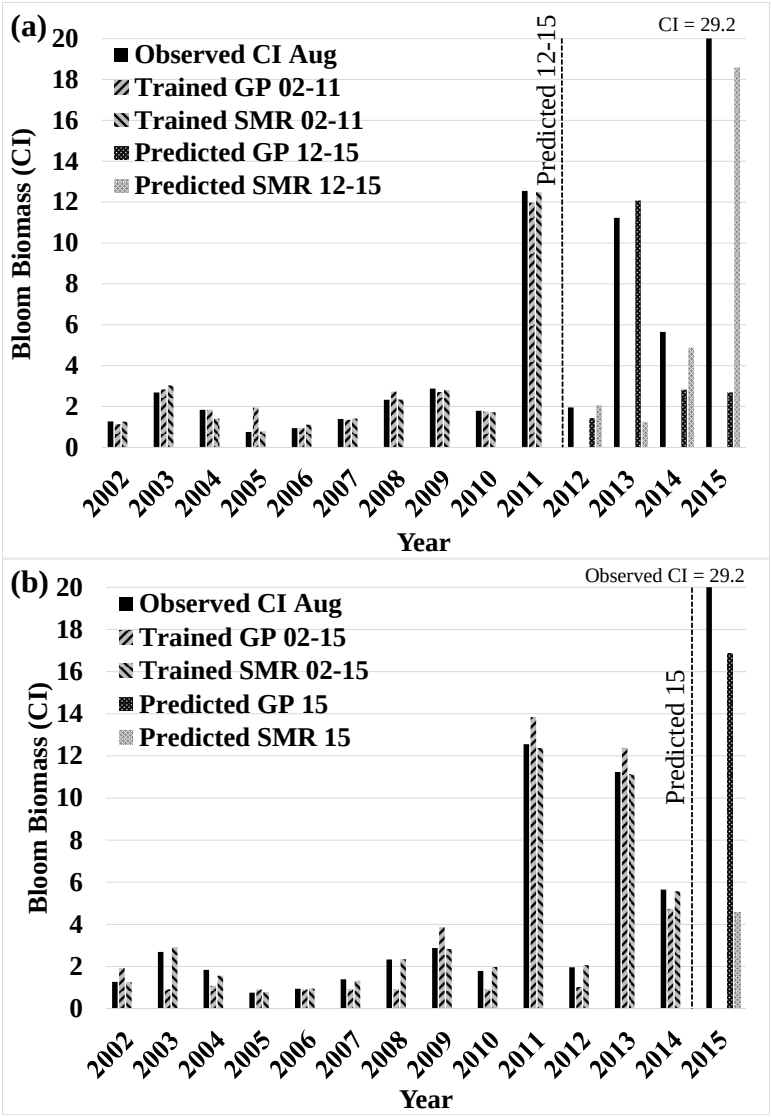


Figure 18. GP and SMR August Results: (a) 2002~2011 Training Period and (b) 2002~2014 Training Period

Model performance of September by two models is compared Figure 19(a). Although GP model underestimates 2012 and 2013, 2014and 2015 are well predicted by the GP model and all predicted CI zones are in agreement with the observed CI zones. Overall SMR predicted CI values are underperformed compare to GP. GP shows better performance in 2015 than SMR and predicted the CI as “significant” same to observation

although the predicted values is 25% less than the observed, while SMR predicted September CI 2015 value 120% more than observed.

In Figure 19(b), GP well predicts with extended training period while SMR model underestimates 2015. The GP model predicts September CI 2015 as "significant" same as the observed CI zone. However, SMR model predicts September CI 2015 as "mild". The September CI value predicted by SMR is 75% less than the observation.

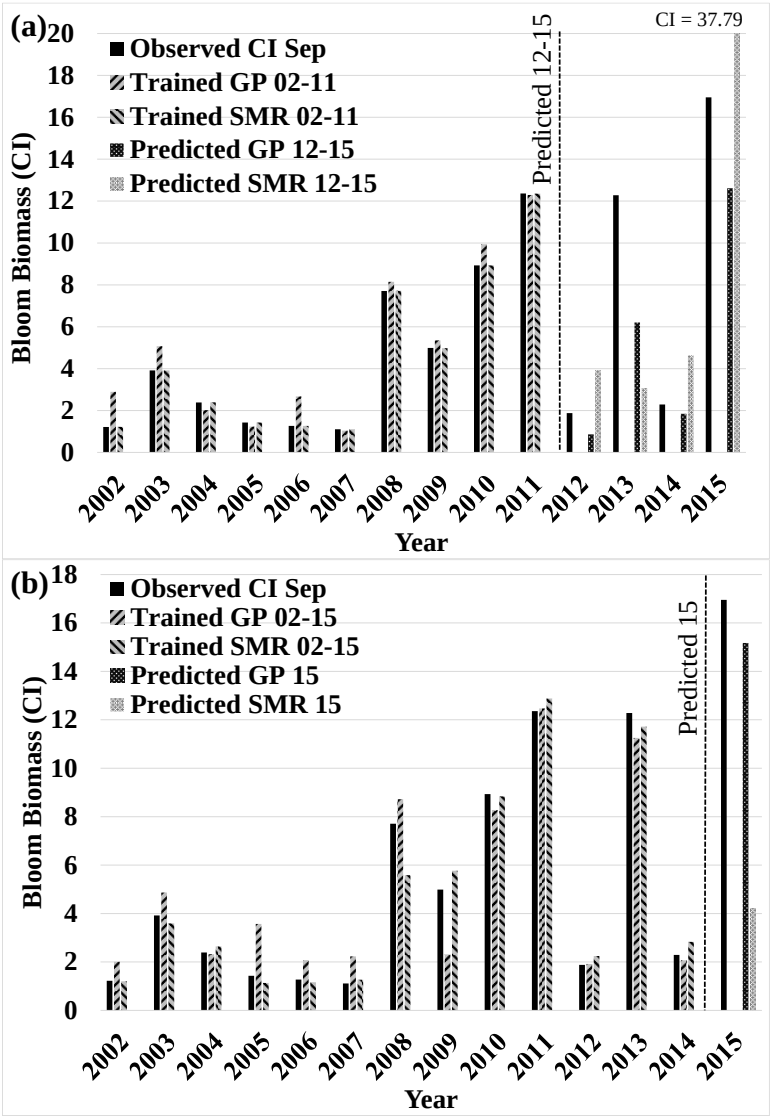


Figure 19. GP and SMR September Results: (a) 2002~2011 Training Period and (b) 2002~2014 Training Period

Model performance of October by two models is compared in Figure 20(a) for training period 2002-2011. Although SMR model underestimates for 2013 and 2014 and overestimates 2012, 2015 is well predicted showing the same “significant” as the observed.

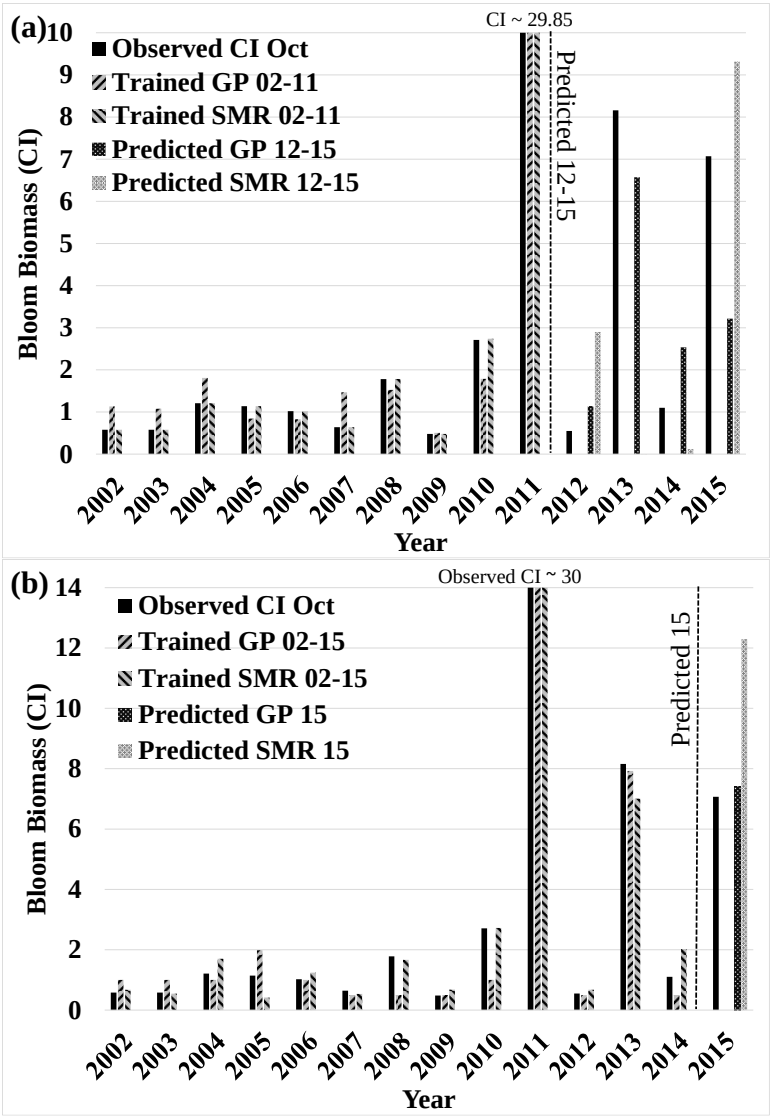


Figure 20. GP and SMR October Results: (a) 2002~2011 Training Period and (b) 2002~2014 Training Period

GP shows overall better performance in 2012-2015 than SMR, except GP predicts 2015 as “mild” which is “significant” in observation. It is noted that SMR predicts October 2015 as “Significant” as observed, which outperforms GP that predicts as “mild”. In Figure 20(b), both models with extended training period well predict October CI 2015 as “significant” same to the observed.

CHAPTER V

CONCLUSION AND FUTURE WORKS

5.1 Summary and Conclusion

HABs are important issues in most fresh water lakes and coastal areas, particularly in Lake Erie. To realize the HABs issue in Lake Erie, extensive literature review was accomplished in this thesis. Various methods or models have been developed for HABs forecasting or prediction and there are still room to improve the performance or operability in HABs prediction in Lake Erie to forecast HABs. To improve the HABs prediction model, widespread analyses and literature review were accomplished on all available variables to select the predictor variables that significantly correlated with the observed HABs events.

The success of machine learning application to natural phenomenon is to select relevant data and to preprocess its scale and sampling period (Brownlee, 2013, Witten, Frank, Hall, & Pal, 2016, Harrington, 2012, Alpaydin, 2014). To exploit the merit of

machine learning fully, all possible data type, its lag times, and averaging periods can be fed to machine learning algorithms. However, preliminary model tests showed that 1) SMR failed training and resulted in a meaningless prediction model and 2) GP failed to converge to terminate optimization due to too many combinations of variables and mathematical functions. Therefore, this study determined to primarily select variables that were commonly used in previous HAB studies. In addition, to train the SMR and GP models efficiently (i.e., reduce the dimension of data set, remove overlapped data, remove rarely correlated data) while exploiting the merit of their learning algorithms, this study screened the preselected variables at a reasonable level using the Spearman nonparametric correlation test that is less sensitive to outliers and thus capable of finding an overall strength and direction between two variables.

The Spearman's rank correlation coefficient was adopted as a standard method to select the significant variables as inputs to two data driven models, SMR and GP. The Spearman selection technique examines up to twenty-eight different lag times and averaging periods for each considered variable. Then, two prediction models, SMR and GP, finally selected the most significant variables by optimizing model parameters that maximize the correlation coefficient between the simulated and the observed (or, minimizing prediction error). Two different training periods were tested to observe if the models predict better when trained for a longer period.

We have discussed and summarized different types of optimization to predict algal bloom index monthly based on different physical parameter such as discharge (*Q*), phosphorous concentration and mass (*TP* and *PM*), soluble reactive phosphorus (*SRP*), nitrogen concentration (*TKN*), water temperature (*Water*), air temperature (*Air*), and wind

speed (*Wind*). Two targets were chosen as: Chlorophyll (*Chl-a*) and Cyanobacteria Index (*CI*) and finally CI selected. Different time lags and average period calculated from the target month up to six months back and finally with Spearman method, some variables selected as final inputs for two different training periods of 2002 to 2011 and 2002 to 2014. Among the tested time steps, biweekly, annual, and monthly, the monthly time step showed the best correlation between the predictor variables and the target variable CI.

First, SMR models were trained for individual months using the selected variables by the Spearman method. The SMR model generated for each month with both from 2002 to 2011 and 2002 to 2014 training period. When the training period increased from 10 years to 13 years, SMR models showed overall 44% improvement in prediction accuracy, where the most improvement was for July and rest of the months did not show substantial improvement.

Second, GP models were trained in a similar manner to SMR. GP also showed the improvement in prediction accuracy about 32.9% by increasing the training period from 10 years to 13 years, where the most improvement was for August, about 120% improvement, and the other months showed a similar range of improvement, average of 3% improvement.

In general, the length of training period is sensitive to the efficiency of model due to the higher chance to include a wider range of training data in a longer training data than a shorter period. Based on the results of SMR and GP, it is found that there is a tradeoff between data length and data quality. Not always the longer training period outperformed over the shorter training period. One of key issues in data-driven machine learning techniques is how to prepare training data to cover the various behaviors of target

variables. By doing so, the trained model can represent the target variables in an acceptable level while avoiding overfitting. Ideally, a longer and wider range of training data is suggested and periodic model re-training is essential to detect the recent HABs mechanism under changing climate and watershed.

The range of CI values (generally from 0 to 30) gives valuable information for decision-making on water quality and watershed management regarding Lake Erie. When considering classes for blooms of HABs, an extreme bloom (exceeding 'significant') can start at a CI of 7. However, a bloom with a CI of 10 compared to 30 can have very different effects on the economy and ecosystem (Stumpf et al., 2016). When the blooms are larger in size they can have devastating effects on the local recreation in the lake as well as for the fisherman (Ho, & Michalak, 2015). For example, cities may see a CI value of 30 and decide to make preparations such as stock piling water in the more possible event of a water treatment plant shut down. The city decision makers may see a CI value of 15 and would still make preparations of a large bloom that is emanate however will take less precautions (Banicki, 2017). In addition, water treatment operators may respond differently to CI value of 10 and CI value of 15, although both CI equal to 10 and 15 considered as 'significant' (Stumpf et al., 2016). Banicki (2017) declares that after the CI equal to 30, fish caught from the western basin of Lake Erie should only be consumed once a week, which means there is a significant different between each value of CI, although all of them are classified as 'significant'.

5.2 Future Research Direction

As both models were trained for maximum 13 years (2002-2014) of historical HAB events, the trained model may generate under- or over-prediction for the unexplained

HAB mechanism in the future. Two suggestions were extracted to handle this issue: 1) develop an extrapolation technique that is statistically sound and operable in the model and 2) test multi-model ensemble approaches to provide most possible HAB prediction. In the future, the finalized SMR and GP models will be coded in a web-based user interface system for Western Lake Erie to help many engineers, decision makers, and public to operate the system easily with clear operational guidance and results interpretation.

REFERENCES

Abdelmutalab, A., Assaleh, K., & El-Tarhuni, M. (2016, April). Automatic modulation classification using hierarchical polynomial classifier and stepwise regression. In *Wireless Communications and Networking Conference (WCNC), 2016 IEEE* (pp. 1-5). IEEE.

Alpaydin, E. (2014). *Introduction to machine learning*. MIT press.

Anderson, D. M., Burkholder, J. M., Cochlan, W. P., Glibert, P. M., Gobler, C. J., Heil, C. A., ... & Trainer, V. L. (2008). Harmful algal blooms and eutrophication: examining linkages from selected coastal regions of the United States. *Harmful Algae*, *8*(1), 39-53.

Anderson, D. M., Glibert, P. M., & Burkholder, J. M. (2002). Harmful algal blooms and eutrophication: nutrient sources, composition, and consequences. *Estuaries and Coasts*, *25*(4), 704-726.

Anderson, D. M., Hoagland, P., Kaoru, Y., & White, A. W. (2000). *Estimated annual economic impacts from harmful algal blooms (HABs) in the United States* (No. WHOI-2000-11). NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION NORMAN OK NATIONAL SEVERE STORMS LAB.

Banicki, J. (2017, July 13). Forecast for Harmful Algal Blooms in Lake Erie in 2017 [Webinar]. In *Ohio Sea Grant.* Retrieved from https://oh-tech.webex.com/oh-tech/onstage/g.php?MTID=e75acc40de0c33e40a90a652945ad280e

Barnett, F. A., Gray, S. T., & Tootle, G. A. (2009). Upper green river basin (United States) streamflow reconstructions. *Journal of Hydrologic Engineering*, *15*(7), 567-579.

Bertani, I., Obenour, D. R., Steger, C. E., Stow, C. A., Gronewold, A. D., & Scavia, D. (2016). Probabilistically assessing the role of nutrient loading in harmful algal bloom formation in western Lake Erie. *Journal of Great Lakes Research*, *42*(6), 1184-1192.

Bingham, M., Sinha, S. K., & Lupi, F. (2015). Economic Benefits of Reducing Harmful Algal Blooms in Lake Erie. *Environmental Consulting & Technology, Inc., Report*, 66.

Brandes, D., Hoffmann, J. G., & Mangarillo, J. T. (2005). Base flow recession rates, low flows, and hydrologic features of small watersheds in Pennsylvania, USA. *JAWRA Journal of the American Water Resources Association*, *41*(5), 1177-1186.

Brownlee, J. (2013). How to Prepare Data for Machine Learning. *Machine Learning Mastery*, *25*.

Burkill, P. H., Mantoura, R. F. C., Llewellyn, C. A., & Owens, N. J. P. (1987). Microzooplankton grazing and selectivity of phytoplankton in coastal waters. *Marine biology*, *93*(4), 581-590.

Calbet, A., & Landry, M. R. (2004). Phytoplankton growth, microzooplankton grazing, and carbon cycling in marine systems. *Limnology and Oceanography*, *49*(1), 51-57.

Chang, K. W., Shen, Y., & Chen, P. C. (2004). Predicting algal bloom in the Techi reservoir using Landsat TM data. *International Journal of Remote Sensing*, *25*(17), 3411-3422.

Chen, Q., Guan, T., Yun, L., Li, R., & Recknagel, F. (2015). Online forecasting chlorophyll a concentrations by an auto-regressive integrated moving average model: Feasibilities and potentials. *Harmful Algae*, *43*, 58-65.

Chen, Y., Shi, R., Shu, S., & Gao, W. (2013). Ensemble and enhanced PM 10 concentration forecast model based on stepwise regression and wavelet analysis. Atmospheric Environment, 74, 346-359.

Cho, S., Lim, B., Jung, J., Kim, S., Chae, H., Park, J., ... & Park, J. K. (2014). Factors affecting algal blooms in a man-made lake and prediction using an artificial neural network. *Measurement*, *53*, 224-233.

Darlington, R. B., & Hayes, A. F. (2016). *Regression analysis and linear models: Concepts, applications, and implementation*. Guilford Publications.

Di Toro, D. M., Thomas, N. A., Herdendorf, C. E., Winfield, R. P., & Connolly, J. P. (1987). A post audit of a Lake Erie eutrophication model. *Journal of great lakes research*, *13*(4), 801-825.

Dudek, G. (2016). Pattern-based local linear regression models for short-term load forecasting. *Electric Power Systems Research*, *130*, 139-147.

Environmental Protection Agency. (2017). *Recommended Binational Phosphorus Targets.* Retrieved June 15, 2017, from https://www.epa.gov/glwqa/recommended-binational-phosphorus-targets.

Faraway, J. J. (2016). *Extending the linear model with R: generalized linear, mixed effects and nonparametric regression models* (Vol. 124). CRC press.

Feng, L., Chen, B., Hayat, T., Alsaedi, A., & Ahmad, B. (2015). Modelling the influence of thermal discharge under wind on algae. *Physics and Chemistry of the Earth, Parts A/B/C*, *79*, 108-114.

Francone, F. D. (1998). Discipulus owner's manual. *Machine Learning Technologies, Inc., Littleton, Colorado*.

Gong, G., Wang, L., Condon, L., Shearman, A., & Lall, U. (2010). A simple framework for incorporating seasonal streamflow forecasts into existing water resource management practices. *JAWRA Journal of the American Water Resources Association*, *46*(3), 574-585.

Great Lakes Monitoring, (2017). *Illinois-Indiana Sea Grant, University of Illinois National Center for Supercomputing Applications and the Department of Civil and Environmental Engineering Data Download.* Retrieved June 01, 2017 from https://greatlakesmonitoring.org/geodashboard/#search

Håkanson, L., Malmaeus, J. M., Bodemer, U., & Gerhardt, V. (2003). Coefficients of variation for chlorophyll, green algae, diatoms, cryptophytes and blue-greens in rivers as a basis for predictive modelling and aquatic management. *Ecological Modelling*, *169*(1), 179-196.

Hallegraeff, G. M. (1993). A review of harmful algal blooms and their apparent global increase. *Phycologia*, *32*(2), 79-99.

Harrington, P. (2012). *Machine learning in action* (Vol. 5). Greenwich, CT: Manning.

Heisler, J., Glibert, P. M., Burkholder, J. M., Anderson, D. M., Cochlan, W., Dennison, W. C., ... & Lewitus, A. (2008). Eutrophication and harmful algal blooms: a scientific consensus. *Harmful algae*, *8*(1), 3-13.

Heuvelmans, G., Muys, B., & Feyen, J. (2006). Regionalisation of the parameters of a hydrological model: Comparison of linear regression models with artificial neural nets. *Journal of Hydrology*, *319*(1), 245-265.

Hitzfeld, B. C., Höger, S. J., & Dietrich, D. R. (2000). Cyanobacterial toxins: removal during drinking water treatment, and human risk assessment. *Environmental health perspectives*, *108*(Suppl 1), 113.

Ho, J. C., & Michalak, A. M. (2015). Challenges in tracking harmful algal blooms: A synthesis of evidence from Lake Erie. *Journal of Great Lakes Research*, *41*(2), 317-325.

Hoagland, P., Anderson, D. M., Kaoru, Y., & White, A. W. (2002). The economic effects of harmful algal blooms in the United States: estimates, assessment issues, and information needs. *Estuaries and Coasts*, *25*(4), 819-837.

Hoeger, S. J., Hitzfeld, B. C., & Dietrich, D. R. (2005). Occurrence and elimination of cyanobacterial toxins in drinking water treatment plants. *Toxicology and applied pharmacology*, *203*(3), 231-242.

Indiana University–Purdue University Indianapolis. (2017). *What causes algal blooms?*. Retrieved February 15, 2017, from http://www.cees.iupui.edu/research/algal-toxicology/bloomfactors.

Irigoien, X., Flynn, K. J., & Harris, R. P. (2005). Phytoplankton blooms: a 'loophole'in microzooplankton grazing impact?. *Journal of Plankton Research*, *27*(4), 313-321.

Jia, Y., Dan, J., Zhang, M., & Kong, F. (2013). Growth characteristics of algae during early stages of phytoplankton bloom in Lake Taihu, China. *Journal of Environmental Sciences*, *25*(2), 254-261.

Joehnk, K. D., Huisman, J. E. F., Sharples, J., Sommeijer, B. E. N., Visser, P. M., & Stroom, J. M. (2008). Summer heatwaves promote blooms of harmful cyanobacteria. *Global change biology*, *14*(3), 495-512.

Kalogirou, S., & Sencan, A. (2010). Artificial intelligence techniques in solar energy applications. In *Solar Collectors and Panels, Theory and Applications*. InTech.

Kang, Y., Koch, F., & Gobler, C. J. (2015). The interactive roles of nutrient loading and zooplankton grazing in facilitating the expansion of harmful algal blooms caused by the pelagophyte, Aureoumbra lagunensis, to the Indian River Lagoon, FL, USA. *Harmful Algae*, *49*, 162-173.

Kim, D. K., Zhang, W., Watson, S., & Arhonditsis, G. B. (2014-a). A commentary on the modelling of the causal linkages among nutrient loading, harmful algal blooms, and hypoxia patterns in Lake Erie. *Journal of Great Lakes Research*, *40*, 117-129.

Kim, K. S., & Park, J. H. (2009). A survey of applications of artificial intelligence algorithms in eco-environmental modelling. *Environmental Engineering Research*, *14*(2), 102-110.

Kim, Y., Shin, H. S., & Plummer, J. D. (2014-b). A wavelet-based autoregressive fuzzy model for forecasting algal blooms. *Environmental Modelling & Software*, *62*, 1-10.

Kozacek, C. (2014, August 2). Toledo issues emergency 'Do Not Drink Water' warning to residents. Retrieved February 5, 2017, from http://www.circleofblue.org/2014/world/toledo-issues-emergency-warning-residents-drink-water/

Lam, D. C. L., Schertzer, W. M., & Fraser, A. S. (1987). Oxygen depletion in Lake Erie: modeling the physical, chemical, and biological interactions, 1972 and 1979. *Journal of Great Lakes Research*, *13*(4), 770-781.

Landsberg, J. H. (2002). The effects of harmful algal blooms on aquatic organisms. *Reviews in Fisheries Science*, *10*(2), 113-390.

Latasa, M., Landry, M. R., Louise, S., & Bidigare, R. R. (1997). Pigment specific growth and grazing rates of phytoplankton in the central equatorial Pacific. *Limnology and Oceanography*, *42*(2), 289-298.

Lee, J. H., & Qu, B. (2004). Hydrodynamic tracking of the massive spring 1998 red tide in Hong Kong. *Journal of Environmental Engineering*, *130*(5), 535-550.

Lee, J. H., Huang, Y., Dickman, M., & Jayawardena, A. W. (2003). Neural network modelling of coastal algal blooms. *Ecological Modelling*, *159*(2), 179-201.

Leon, L. F., Smith, R. E., Hipsey, M. R., Bocaniov, S. A., Higgins, S. N., Hecky, R. E., ... & Guildford, S. J. (2011). Application of a 3D hydrodynamic–biological model for seasonal and spatial dynamics of water quality and phytoplankton in Lake Erie. *Journal of Great Lakes Research*, *37*(1), 41-53.

Lou, I., Xie, Z., Ung, W. K., & Mok, K. M. (2015). Integrating Support Vector Regression with Particle Swarm Optimization for numerical modeling for algal blooms of freshwater. *Applied Mathematical Modelling*, *39*(19), 5907-5916.

Lui, G. C., Li, W. K., Leung, K. M., Lee, J. H., & Jayawardena, A. W. (2007). Modelling algal blooms using vector autoregressive model with exogenous variables and long memory filter. *Ecological modelling*, *200*(1), 130-138.

Maier, H. R., Sayed, T., & Lence, B. J. (2001). Forecasting cyanobacterium Anabaena spp. in the River Murray, South Australia, using B-spline neurofuzzy models. *Ecological Modelling*, *146*(1), 85-96.

Michalak, A. M., Anderson, E. J., Beletsky, D., Boland, S., Bosch, N. S., Bridgeman, T. B., ... & DePinto, J. V. (2013). Record-setting algal bloom in Lake Erie caused by agricultural and meteorological trends consistent with expected future conditions. *Proceedings of the National Academy of Sciences*, *110*(16), 6448-6452.

Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.). (2013). *Machine learning: An artificial intelligence approach*. Springer Science & Business Media.

Modigh, M., & Franzè, G. (2009). Changes in phytoplankton and microzooplankton populations during grazing experiments at a Mediterranean coastal site. *Journal of Plankton Research*, *31*(8), 853-864.

Muttil, N., & Chau, K. W. (2006). Neural network and genetic programming for modelling coastal algal blooms. *International Journal of Environment and Pollution*, *28*(3-4), 223-238.

Muttil, N., & Lee, J. H. (2005). Genetic programming for analysis and real-time prediction of coastal algal blooms. *Ecological modelling*, *189*(3), 363-376.

National Center for Water Quality Research. (2017). *Heidelberg University, Tributary Data Download.* Retrieved June 01, 2017 from https://www.heidelberg.edu/academics/research-and-centers/national-center-for-water-quality-research/tributary-data-download

Obenour, D. R., Gronewold, A. D., Stow, C. A., & Scavia, D. (2014). Using a Bayesian hierarchical model to improve Lake Erie cyanobacteria bloom forecasts. *Water Resources Research*, *50*(10), 7847-7860.

Ohio Department of Health. (2016). *Harmful Algal Blooms*. Retrieved June 15, 2017, from https://www.odh.ohio.gov/odhprograms/eh/HABs/algalblooms.aspx.

Ohio Department of Natural Resources (ODNR). (2017). *Lake Erie Facts.* Retrieved June 15, 2017, from http://www.dnr.state.oh.us/tabid/7828/default.aspx.

Paerl, H. W., & Huisman, J. (2009). Climate change: a catalyst for global expansion of harmful cyanobacterial blooms. *Environmental microbiology reports*, *1*(1), 27-37.

Peña-Arancibia, J. L., Van Dijk, A. I. J. M., Mulligan, M., & Bruijnzeel, L. A. (2010). The role of climatic and terrain attributes in estimating baseflow recession in tropical catchments. *Hydrology and Earth System Sciences*, *14*(11), 2193-2205.

Persaud, A. D., Paterson, A. M., Dillon, P. J., Winter, J. G., Palmer, M., & Somers, K. M. (2015). Forecasting cyanobacteria dominance in Canadian temperate lakes. *Journal of environmental management*, *151*, 343-352.

Qin, B., Li, W., Zhu, G., Zhang, Y., Wu, T., & Gao, G. (2015). Cyanobacterial bloom management through integrated monitoring and forecasting in large shallow eutrophic Lake Taihu (China). *Journal of hazardous materials*, *287*, 356-363.

Qin, B., Zhu, G., Gao, G., Zhang, Y., Li, W., Paerl, H. W., & Carmichael, W. W. (2010). A drinking water crisis in Lake Taihu, China: linkage to climatic variability and lake management. *Environmental management*, *45*(1), 105-112.

Rajaee, T., & Boroumand, A. (2015). Forecasting of chlorophyll-a concentrations in South San Francisco Bay using five different models. *Applied Ocean Research*, *53*, 208-217.

Ramsami, P., & Oree, V. (2015). A hybrid method for forecasting the energy output of photovoltaic systems. *Energy Conversion and Management*, *95*, 406-413.

Recknagel, F., Bobbin, J., Whigham, P., & Wilson, H. (2002). Comparative application of artificial neural networks and genetic algorithms for multivariate time-series modelling of algal blooms in freshwater lakes. *Journal of Hydroinformatics*, *4*(2), 125-133.

Recknagel, F., French, M., Harkonen, P., & Yabunaka, K. I. (1997). Artificial neural network approach for modelling and prediction of algal blooms. *Ecological Modelling*, *96*(1-3), 11-28.

Reutter, J., & Dierkes, C. (2014, August 5). *Harmful Algal Bloom Q&A and Updates*. Retrieved June 01, 2017, from https://ohioseagrant.osu.edu/news/2014/a8990/harmful-algal-bloom-qa-updates

Sayegh, A. S., Munir, S., & Habeebullah, T. M. (2014). Comparing the performance of statistical models for predicting PM10 concentrations. *Aerosol and Air Quality Research*, *14*(3), 653-65.

Scavia, D., Allan, J. D., Arend, K. K., Bartell, S., Beletsky, D., Bosch, N. S., ... & Dolan, D. M. (2014). Assessing and addressing the re-eutrophication of Lake Erie: Central basin hypoxia. *Journal of Great Lakes Research*, *40*(2), 226-246.

Schottler, S. P., Eisenreich, S. J., & Capel, P. D. (1994). Atrazine, alachlor, and cyanazine in a large agricultural river system. *Environmental science & technology*, *28*(6), 1079-1089.

Sharma, M. J., & Yu, S. J. (2015). Stepwise regression data envelopment analysis for variable reduction. *Applied Mathematics and Computation*, *253*, 126-134.

Sheng, H., Liu, H., Wang, C., Guo, H., Liu, Y., & Yang, Y. (2012). Analysis of cyanobacteria bloom in the Waihai part of Dianchi Lake, China. *Ecological Informatics*, *10*, 37-48.

Sitoki, L., Kurmayer, R., & Rott, E. (2012). Spatial variation of phytoplankton composition, biovolume, and resulting microcystin concentrations in the Nyanza Gulf (Lake Victoria, Kenya). *Hydrobiologia*, *691*(1), 109-122.

Sivapragasam, C., Muttil, N., Muthukumar, S., & Arun, V. M. (2010). Prediction of algal blooms using genetic programming. *Marine pollution bulletin*, *60*(10), 1849-1855.

Smayda, T. J. (2008). Complexity in the eutrophication–harmful algal bloom relationship, with comment on the importance of grazing. *Harmful Algae*, *8*(1), 140-151.

Stocker, T. F., Qin, D., Plattner, G. K., Tignor, M., Allen, S. K., Boschung, J., ... & Midgley, P. M. (2013). Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change, 1535 pp.

Stumpf, R. P., Davis, T. W., Wynne, T. T., Graham, J. L., Loftin, K. A., Johengen, T. H., ... & Burtner, A. (2016). Challenges for mapping cyanotoxin patterns from remote sensing of cyanobacteria. *Harmful Algae*, *54*, 160-173.

Stumpf, R. P., Johnson, L. T., Wynne, T. T., & Baker, D. B. (2016). Forecasting annual cyanobacterial bloom biomass to inform management decisions in Lake Erie. *Journal of Great Lakes Research*, *42*(6), 1174-1183.

Stumpf, R. P., Wynne, T. T., Baker, D. B., & Fahnenstiel, G. L. (2012). Interannual variability of cyanobacterial blooms in Lake Erie. *PloS one*, *7*(8), e42444.

Talib, A., Recknagel, F., Cao, H., & van der Molen, D. T. (2008). Forecasting and explanation of algal dynamics in two shallow lakes by recurrent artificial neural network and hybrid evolutionary algorithm. *Mathematics and Computers in Simulation*, *78*(2), 424-434.

Taranu, Z. E., Gregory-Eaves, I., Steele, R. J., Beaulieu, M., & Legendre, P. (2017). Predicting microcystin concentrations in lakes and reservoirs at a continental scale: A new framework for modelling an important health risk factor. *Global Ecology and Biogeography*, *26*(6), 625-637.

Thornton, J. A., Harding, W. R., Dent, M., Hart, R. C., Lin, H., Rast, C. L., ... & Slawski, T. M. (2013). Eutrophication as a 'wicked'problem. *Lakes & reservoirs: Research & management*, *18*(4), 298-316.

Ul-Saufie, A., Yahya, A., Ramli, N., & Hamid, H. (2012). Future PM10 concentration prediction using quantile regression models. In *International Conference on Environmental and Agriculture Engineering, IACSIT Press, Singapore* (Vol. 37).

United States Department of Agriculture. (2005). Western Lake Erie Basin Water Resources Protection Plan. USDA, Washington D.C. 1–24.

United States Geological Survey government agency. (2017). *USGS Surface-Water Monthly Statistics for the Nation.* Retrieved June 01, 2017 from https://waterdata.usgs.gov/nwis/monthly/?search_site_no=04193500&agency_cd=USGS&referred_module=sw&format=sites_selection_links

Van Dolah, F. M. (2005). Effects of harmful algal blooms. *Marine mammal research: conservation beyond crisis (JE Reynolds III, WF Perrin, RR Reeves, S.*

Montgomery, and TJ Ragen, eds.). Johns Hopkins University Press, Baltimore, Maryland, 85-99.

Wells, M. L., Trainer, V. L., Smayda, T. J., Karlson, B. S., Trick, C. G., Kudela, R. M., ... & Cochlan, W. P. (2015). Harmful algal blooms and climate change: Learning from the past and present to forecast the future. *Harmful Algae*, *49*, 68-93.

Whigham, P. A., & Recknagel, F. (1999). Predictive modelling of plankton dynamics in freshwater lakes using genetic programming. In: MODSIM International Congress on Modelling and Simulation, Modeling and Simulation Society of Australia and New Zealand, 6–9 December, 1999, New Zealand, pp. 691–696.

Whitley, D. (2014). An executable model of a simple genetic algorithm. *Foundations of genetic algorithms*, *2*(1519), 45-62.

Wilson, C., Wright, E., Bronnenhuber, J., MacDonald, F., Belore, M., & Locke, B. (2014). Tracking ghosts: combined electrofishing and environmental DNA surveillance efforts for Asian carps in Ontario waters of Lake Erie. *Management of Biological Invasions*, *5*(3), 225-231.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Wu, T., Qin, B., Brookes, J. D., Shi, K., Zhu, G., Zhu, M., ... & Wang, Z. (2015). The influence of changes in wind patterns on the areal extension of surface cyanobacterial blooms in a large shallow lake in China. *Science of The Total Environment*, *518*, 24-30.

Zhang, H., Culver, D. A., & Boegman, L. (2008). A two-dimensional ecological model of Lake Erie: application to estimate dreissenid impacts on large lake plankton populations. *ecological modelling*, *214*(2), 219-241.

Zhang, H., Hu, W., Gu, K., Li, Q., Zheng, D., & Zhai, S. (2013). An improved ecological model and software for short-term algal bloom forecasting. *Environmental modelling & software*, *48*, 152-162.

APPENDICES

APPENDIX A: C language output of GP models for training period 2002 to 2011

GP MODEL FOR JULY

```
#include <math.h>
#include <float.h>
#define TRUNC(x)(((x)>=0) ? floor(x) : ceil(x))
#define C_FPREM (_finite(f[0]/f[1]) ? f[0]-(TRUNC(f[0]/f[1])*f[1]) : f[0]/f[1])
#define C_F2XM1 (((fabs(f[0])<=1) && (!_isnan(f[0]))) ? (pow(2,f[0])-1) :
((!_finite(f[0]) && !_isnan(f[0]) && (f[0]<0)) ? -1 : f[0]))

float DiscipulusCFunction(float v[])
{
 long double f[8];
 long double tmp = 0;
 int cflag = 0;

 f[0]=f[1]=f[2]=f[3]=f[4]=f[5]=f[6]=f[7]=0;

 double Q5,1=v[0] ;
 double Q1,6=v[1] ;
 double Pm1,5=v[2] ;
 double Tkn3,3=v[3] ;
 double Tkn1,6=v[4] ;

 L0:   f[0]-=-1.924433708190918f;
 L1:   f[0]-=-1.174947738647461f;
 L2:   f[0]*=Tkn1,6;
 L3:   f[0]=-f[0];
 L4:   f[0]/=Q5,1;
 L5:   f[3]+=f[0];
 L6:   f[0]=fabs(f[0]);
 L7:   f[0]+=Pm1,5;
 L8:   f[0]+=1.987620830535889f;
 L9:   f[0]+=f[0];
 L10:  f[0]=cos(f[0]);
 L11:  f[3]/=f[0];
 L12:  f[0]=cos(f[0]);
 L13:  f[0]=sin(f[0]);
 L14:  f[0]+=f[3];
 L15:  f[0]=-f[0];
 L16:  f[0]+=1.77857518196106f;
```

```
 L17:

 if (!_finite(f[0])) f[0]=0;

 return f[0];
}
float DiscipulusCRegressionFunction(float  v [])
{
   float ret = DiscipulusCFunction(v) ;
   return ret;
}
```

GP MODEL FOR AUGUST

```
#include <math.h>
#include <float.h>
#define TRUNC(x)(((x)>=0) ? floor(x) : ceil(x))
#define C_FPREM (_finite(f[0]/f[1]) ? f[0]-(TRUNC(f[0]/f[1])*f[1]) : f[0]/f[1])
#define C_F2XM1 (((fabs(f[0])<=1) && (!_isnan(f[0]))) ? (pow(2,f[0])-1) :
((!_finite(f[0]) && !_isnan(f[0]) && (f[0]<0)) ? -1 : f[0]))

float DiscipulusCFunction(float v[])
{
 long double f[8];
 long double tmp = 0;
 int cflag = 0;

 f[0]=f[1]=f[2]=f[3]=f[4]=f[5]=f[6]=f[7]=0;

 double Q1,6=v[0] ;
 double Pm5,1=v[1] ;
 double Pm1,6=v[2] ;
 double Tkn1,6=v[3] ;
 double Water1,3=v[4] ;
 double Water1,6=v[5] ;

 L0:   f[0]+=-0.8637528419494629f;
 L1:   f[0]-=0.6909141540527344f;
 L2:   f[0]*=Water1,6;
 L3:   f[0]=cos(f[0]);
 L4:   f[0]=cos(f[0]);
 L5:   f[1]-=f[0];
 L6:   f[0]+=Water1,3;
 L7:   f[0]-=Pm1,6;
 L8:   f[0]+=Water1,3;
 L9:   f[0]=cos(f[0]);
 L10:  f[1]-=f[0];
```

```
 L11:  f[0]*=pow(2,TRUNC(f[1]));
 L12:  f[0]=fabs(f[0]);
 L13:  f[0]*=pow(2,TRUNC(f[1]));
 L14:  f[0]*=Tkn1,6;
 L15:  f[0]/=0.1240770965814591f;
 L16:

 if (!_finite(f[0])) f[0]=0;

 return f[0];
}
float DiscipulusCRegressionFunction(float  v [])
{
  float ret = DiscipulusCFunction(v) ;
  return ret;
}
```

GP MODEL FOR SEPTEMBER

```
#include <math.h>
#include <float.h>
#define TRUNC(x)(((x)>=0) ? floor(x) : ceil(x))
#define C_FPREM (_finite(f[0]/f[1]) ? f[0]-(TRUNC(f[0]/f[1])*f[1]) : f[0]/f[1])
#define C_F2XM1 (((fabs(f[0])<=1) && (!_isnan(f[0]))) ? (pow(2,f[0])-1) :
((!_finite(f[0]) && !_isnan(f[0]) && (f[0]<0)) ? -1 : f[0]))

float DiscipulusCFunction(float v[])
{
 long double f[8];
 long double tmp = 0;
 int cflag = 0;

 f[0]=f[1]=f[2]=f[3]=f[4]=f[5]=f[6]=f[7]=0;

 double Q3,3=v[0] ;
 double Q1,6=v[1] ;
 double TP3,4=v[2] ;
 double Pm1,6=v[3] ;
 double Tkn1,1=v[4] ;
 double Tkn3,4=v[5] ;

 L0:   f[0]+=2.203821659088135f;
 L1:   f[0]=sin(f[0]);
 L2:   f[0]-=-0.8215113878250122f;
 L3:   f[0]-=Q1,6;
 L4:   f[0]+=Tkn3,4;
 L5:   f[0]=sin(f[0]);
```

```
 L6:   f[0]+=4.120148658752441f;
 L7:   f[0]*=Tkn1,1;
 L8:   f[0]*=Pm1,6;
 L9:   f[0]+=-2.35340166091919f;
 L10:  f[0]-=-1.76183032989502f;
 L11:  f[0]=sin(f[0]);
 L12:  f[0]-=-0.7284336686134338f;
 L13:  f[0]-=Q1,6;
 L14:  f[1]-=f[0];
 L15:  f[0]=sin(f[0]);
 L16:  f[1]*=f[0];
 L17:  f[0]-=-2.294054746627808f;
 L18:  f[0]*=Tkn1,1;
 L19:  f[0]/=TP3,4;
 L20:  f[0]+=-0.67620849609375f;
 L21:  f[0]*=Pm1,6;
 L22:  f[1]/=f[0];
 L23:  f[0]-=f[1];
 L24:  f[0]=fabs(f[0]);
 L25:  f[0]*=Tkn3,4;
 L26:

 if (!_finite(f[0])) f[0]=0;

 return f[0];
}
float DiscipulusCRegressionFunction(float  v [])
{
  float ret = DiscipulusCFunction(v) ;
  return ret;
}
```

GP MODEL FOR OCTOBER

```
#include <math.h>
#include <float.h>
#define TRUNC(x)(((x)>=0) ? floor(x) : ceil(x))
#define C_FPREM (_finite(f[0]/f[1]) ? f[0]-(TRUNC(f[0]/f[1])*f[1]) : f[0]/f[1])
#define C_F2XM1 (((fabs(f[0])<=1) && (!_isnan(f[0]))) ? (pow(2,f[0])-1) :
((!_finite(f[0]) && !_isnan(f[0]) && (f[0]<0)) ? -1 : f[0]))

float DiscipulusCFunction(float v[])
{
 long double f[8];
 long double tmp = 0;
 int cflag = 0;
```

```
 f[0]=f[1]=f[2]=f[3]=f[4]=f[5]=f[6]=f[7]=0;

 double Q5,2=v[0] ;
 double Q1,6=v[1] ;
 double TP1,6=v[2] ;
 double PM5,2=v[3] ;
 double PM1,6=v[4] ;
 double Wind3,2=v[5] ;
 double Wind1,4=v[6] ;

 L0:    f[0]+=Wind1,4;
 L1:    f[1]+=f[0];
 L2:    f[1]*=f[0];
 L3:    f[0]=cos(f[0]);
 L4:    f[0]+=f[0];
 L5:    f[0]+=-0.1401152610778809f;
 L6:    f[0]=-f[0];
 L7:    f[1]+=f[0];
 L8:    f[1]*=f[0];
 L9:    f[0]=sin(f[0]);
 L10:   f[0]*=f[1];
 L11:   f[1]/=f[0];
 L12:   f[0]=sin(f[0]);
 L13:   f[0]*=f[1];
 L14:   f[1]/=f[0];
 L15:   f[0]*=f[0];
 L16:   f[0]+=1.744837045669556f;
 L17:   f[0]/=f[1];
 L18:   f[0]=-f[0];
 L19:   f[0]+=Q1,6;
 L20:   f[0]=sin(f[0]);
 L21:   f[0]*=f[1];
 L22:   f[0]=fabs(f[0]);
 L23:   f[0]+=TP1,6;
 L24:   f[0]+=TP1,6;
 L25:
 if (!_finite(f[0])) f[0]=0;
 return f[0];
}
float DiscipulusCRegressionFunction(float  v [])
{
  float ret = DiscipulusCFunction(v) ;
  return ret;
}
```

APPENDIX B: C language output of GP models for training period 2002 to 2014

GP MODEL FOR JULY

```
#include <math.h>
#include <float.h>
#define TRUNC(x)(((x)>=0) ? floor(x) : ceil(x))
#define C_FPREM (_finite(f[0]/f[1]) ? f[0]-(TRUNC(f[0]/f[1])*f[1]) : f[0]/f[1])
#define C_F2XM1 (((fabs(f[0])<=1) && (!_isnan(f[0]))) ? (pow(2,f[0])-1) :
((!_finite(f[0]) && !_isnan(f[0]) && (f[0]<0)) ? -1 : f[0]))

float DiscipulusCFunction(float v[])
{
 long double f[8];
 long double tmp = 0;
 int cflag = 0;

 f[0]=f[1]=f[2]=f[3]=f[4]=f[5]=f[6]=f[7]=0;

 double Q2,2=v[0] ;
 double Q1,5=v[1] ;
 double P3,1=v[2] ;
 double P1,5=v[3] ;
 double Pm3,1=v[4] ;
 double Pm1,5=v[5] ;
 double Tkn1,6=v[6] ;
 double Water3,4=v[7] ;

 L0:   f[0]-=0.9177978038787842f;
 L1:   f[0]-=Pm3,1;
 L2:   f[0]=fabs(f[0]);
 L3:   f[1]+=f[0];
 L4:   f[0]=cos(f[0]);
 L5:   f[0]+=-1.360518217086792f;
 L6:   f[0]+=f[1];
 L7:   f[0]=sin(f[0]);
 L8:   f[0]*=f[0];
 L9:   f[0]=sin(f[0]);
 L10:  f[0]+=f[0];
 L11:  f[1]/=f[0];
 L12:  f[0]-=f[0];
 L13:  f[0]-=Pm3,1;
 L14:  f[0]=fabs(f[0]);
 L15:  f[0]+=f[0];
 L16:  f[1]/=f[0];
 L17:  f[1]/=f[0];
```

```
 L18:  f[0]-=Q2,2;
 L19:  f[0]=sin(f[0]);
 L20:  f[0]=fabs(f[0]);
 L21:  f[0]*=1.084159851074219f;
 L22:  f[0]+=Tkn1,6;
 L23:  f[0]+=-1.364008665084839f;
 L24:  f[0]+=f[1];
 L25:  f[0]*=f[0];
 L26:

 if (!_finite(f[0])) f[0]=0;

 return f[0];
}
float DiscipulusCRegressionFunction(float  v [])
{
   float ret = DiscipulusCFunction(v) ;
   return ret;
}
```

GP MODEL FOR AUGUST

```
#include <math.h>
#include <float.h>
#define TRUNC(x)(((x)>=0) ? floor(x) : ceil(x))
#define C_FPREM (_finite(f[0]/f[1]) ? f[0]-(TRUNC(f[0]/f[1])*f[1]) : f[0]/f[1])
#define C_F2XM1 (((fabs(f[0])<=1) && (!_isnan(f[0]))) ? (pow(2,f[0])-1) :
((!_finite(f[0]) && !_isnan(f[0]) && (f[0]<0)) ? -1 : f[0]))

float DiscipulusCFunction(float v[])
{
 long double f[8];
 long double tmp = 0;
 int cflag = 0;

 f[0]=f[1]=f[2]=f[3]=f[4]=f[5]=f[6]=f[7]=0;

 double Q5,1=v[0] ;
 double Q4,2=v[1] ;
 double Q1,6=v[2] ;
 double Pm5,1=v[3] ;
 double Pm4,2=v[4] ;
 double Pm1,6=v[5] ;
 double Water2,1=v[6] ;

 L0:   f[0]+=Q5,1;
 L1:   f[0]-=Pm5,1;
```

```
 L2:    f[0]/=0.7233922481536865f;
 L3:    f[0]/=Q4,2;
 L4:    f[0]*=f[0];
 L5:    f[0]-=-1.196667432785034f;
 L6:    f[0]+=Q1,6;
 L7:    f[0]/=0.7233922481536865f;
 L8:    f[0]/=Q4,2;
 L9:    f[0]*=f[0];
 L10:   f[0]*=f[0];
 L11:   f[0]+=f[0];
 L12:   f[0]*=f[0];
 L13:   f[0]*=f[0];
 L14:   f[0]/=Pm4,2;
 L15:   f[0]+=Q5,1;
 L16:   f[0]-=Pm5,1;
 L17:   f[0]/=0.7233922481536865f;
 L18:   f[0]/=Q4,2;
 L19:   f[0]*=f[0];
 L20:   f[0]*=f[0];
 L21:   f[0]*=f[0];
 L22:   f[0]+=Q1,6;
 L23:   f[0]/=0.7233922481536865f;
 L24:   f[0]/=Q4,2;
 L25:   f[0]*=f[0];
 L26:   f[0]-=-1.364777803421021f;
 L27:   f[0]-=Pm5,1;
 L28:   f[0]=cos(f[0]);
 L29:   f[0]*=f[0];
 L30:   f[0]+=f[0];
 L31:   f[0]*=f[0];
 L32:   f[0]*=f[0];
 L33:   f[0]+=0.9177978038787842f;
 L34:

 if (!_finite(f[0])) f[0]=0;

 return f[0];
}
float DiscipulusCRegressionFunction(float  v [])
{
  float ret = DiscipulusCFunction(v) ;
  return ret;
}
```

GP MODEL FOR SEPTEMBER

```
#include <math.h>
#include <float.h>
#define TRUNC(x)(((x)>=0) ? floor(x) : ceil(x))
#define C_FPREM (_finite(f[0]/f[1]) ? f[0]-(TRUNC(f[0]/f[1])*f[1]) : f[0]/f[1])
#define C_F2XM1 (((fabs(f[0])<=1) && (!_isnan(f[0]))) ? (pow(2,f[0])-1) :
((!_finite(f[0]) && !_isnan(f[0]) && (f[0]<0)) ? -1 : f[0]))

float DiscipulusCFunction(float v[])
{
 long double f[8];
 long double tmp = 0;
 int cflag = 0;

 f[0]=f[1]=f[2]=f[3]=f[4]=f[5]=f[6]=f[7]=0;

 double Q3,1=v[0] ;
 double Q6,1=v[1] ;
 double Q3,3=v[2] ;
 double Q1,6=v[3] ;
 double Pm3,3=v[4] ;
 double Pm1,6=v[5] ;
 double Tkn3,4=v[6] ;

 L0:   f[0]=cos(f[0]);
 L1:   f[0]+=Q3,1;
 L2:   f[0]+=-1.549970149993897f;
 L3:   f[0]=sin(f[0]);
 L4:   f[0]*=f[0];
 L5:   f[0]*=f[0];
 L6:   f[0]*=f[0];
 L7:   f[3]+=f[0];
 L8:   f[0]+=f[3];
 L9:   f[0]*=f[0];
 L10:  f[0]*=Pm3,3;
 L11:  f[0]*=0.03275442123413086f;
 L12:  f[0]-=-1.259177207946777f;
 L13:  f[0]*=Tkn3,4;
 L14:  f[0]*=1.086833715438843f;
 L15:  f[0]*=Tkn3,4;
 L16:  f[0]*=Tkn3,4;
 L17:  f[0]=sqrt(f[0]);
 L18:

 if (!_finite(f[0])) f[0]=0;
```

```
  return f[0];
}
float DiscipulusCRegressionFunction(float  v [])
{
   float ret = DiscipulusCFunction(v) ;
   return ret;
}
```

GP MODEL FOR OCTOBER

```
#include <math.h>
#include <float.h>
#define TRUNC(x)(((x)>=0) ? floor(x) : ceil(x))
#define C_FPREM (_finite(f[0]/f[1]) ? f[0]-(TRUNC(f[0]/f[1])*f[1]) : f[0]/f[1])
#define C_F2XM1 (((fabs(f[0])<=1) && (!_isnan(f[0]))) ? (pow(2,f[0])-1) :
((!_finite(f[0]) && !_isnan(f[0]) && (f[0]<0)) ? -1 : f[0]))

float DiscipulusCFunction(float v[])
{
  long double f[8];
  long double tmp = 0;
  int cflag = 0;

  f[0]=f[1]=f[2]=f[3]=f[4]=f[5]=f[6]=f[7]=0;

  double Q5,2=v[0] ;
  double Q1,6=v[1] ;
  double TP1,6=v[2] ;
  double PM5,2=v[3] ;
  double PM1,6=v[4] ;
  double Air3,4=v[5] ;
  double Wind3,2=v[6] ;
  double Wind1,4=v[7] ;

  L0:    f[0]+=0.0026216506958007 81f;
  L1:    f[0]*=f[0];
  L2:    f[0]+=1.505081653594971f;
  L3:    f[0]*=PM5,2;
  L4:    f[0]=cos(f[0]);
  L5:    f[0]+=f[0];
  L6:    f[0]+=f[0];
  L7:    f[0]=sin(f[0]);
  L8:    f[0]=fabs(f[0]);
  L9:    f[0]*=-0.9765300750732422f;
  L10:   f[0]-=PM1,6;
```

```
 L11:  f[0]-=Wind3,2;
 L12:  f[0]=cos(f[0]);
 L13:  f[0]+=f[0];
 L14:  f[0]+=f[0];
 L15:  f[0]=fabs(f[0]);
 L16:  f[0]*=1.77857518196106f;
 L17:  f[1]-=f[0];
 L18:  f[0]/=PM5,2;
 L19:  f[0]-=Wind1,4;
 L20:  f[1]-=f[0];
 L21:  f[0]*=pow(2,TRUNC(f[1]));
 L22:  f[0]/=PM5,2;
 L23:  f[0]=cos(f[0]);
 L24:  f[0]*=pow(2,TRUNC(f[1]));
 L25:

 if (!_finite(f[0])) f[0]=0;

 return f[0];
}
float DiscipulusCRegressionFunction(float  v [])
{
  float ret = DiscipulusCFunction(v) ;
  return ret;
}
```

APPENDIX C: Step by Step User Guide to Run Discipulus

How to Apply the Inputs and Run the GP

Every time you start Discipulus, the Project Setup Wizard comes up automatically (Francone, 1998). Figure C-1 shows Discipulus right after it has started. The first page of the Project Setup Wizard is showing:

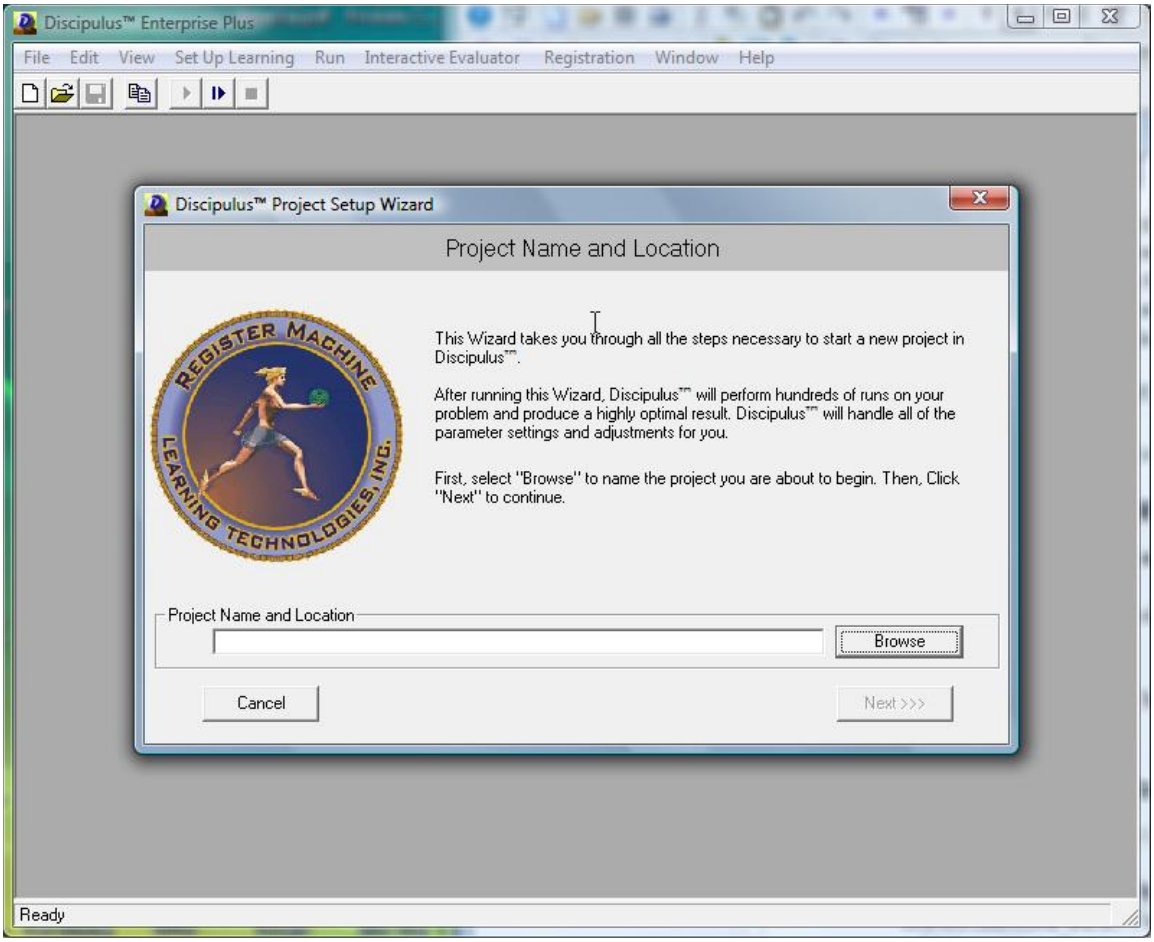


Figure C-1. The Project Name and Location Window of the Project Wizard

The project setup wizard takes you through five simple steps:

1. Name and Save Your Project.

When you first start the project wizard, you see the Project Name and Location Window. Click on the Browse button like Figure C-2 to select a folder and name for your project file. The project file stores all information about the project you will run.

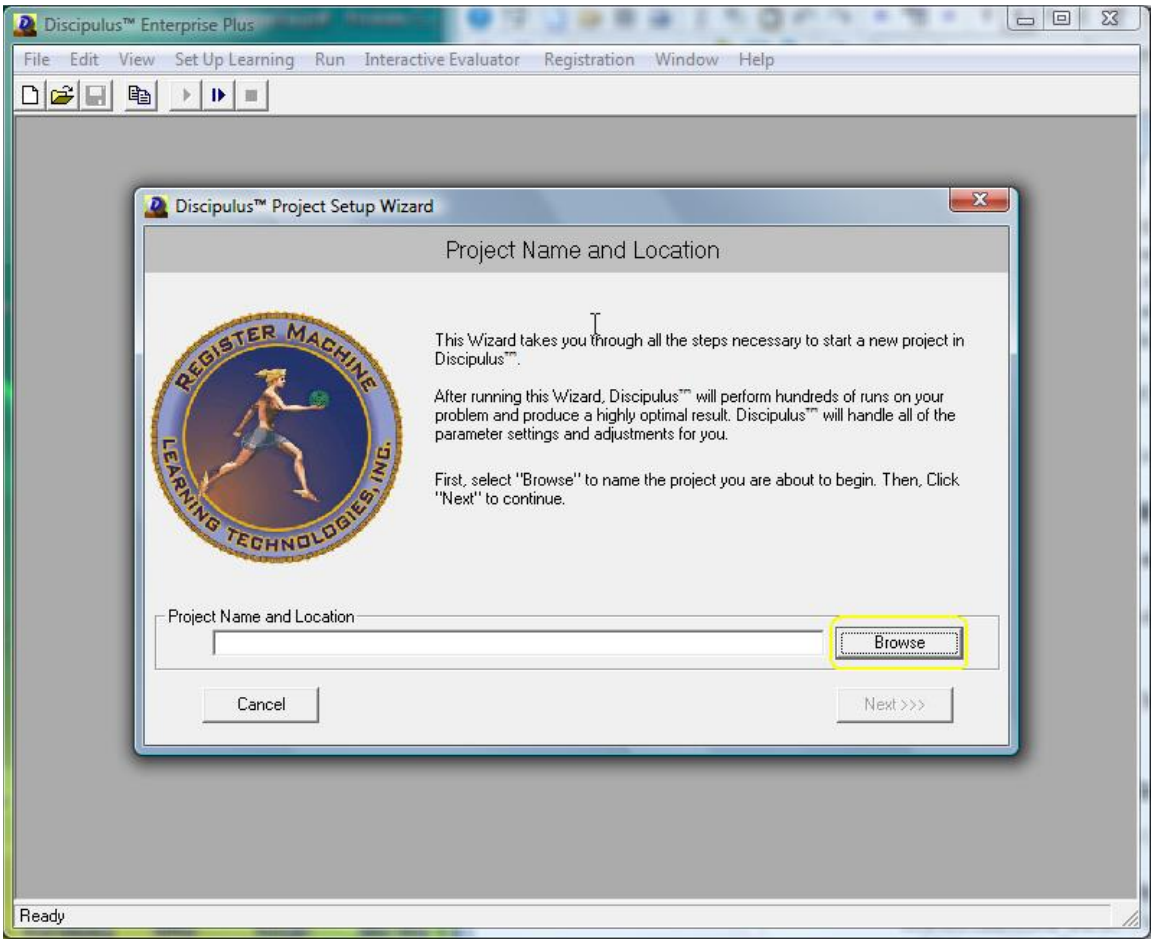


Figure C-2. The Project Name and Location Window. Click on Browse to Name and Locate your Project File

2. **Select Data for Training**.

The second window in the project wizard lets you tell Discipulus how and where to get data for training. That window is shown in Figure C-3.

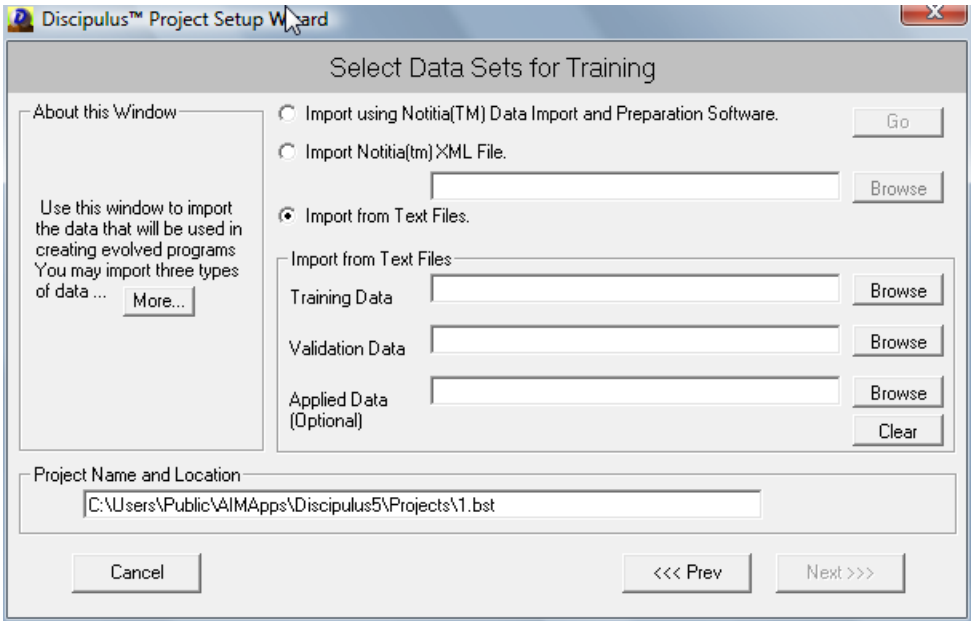
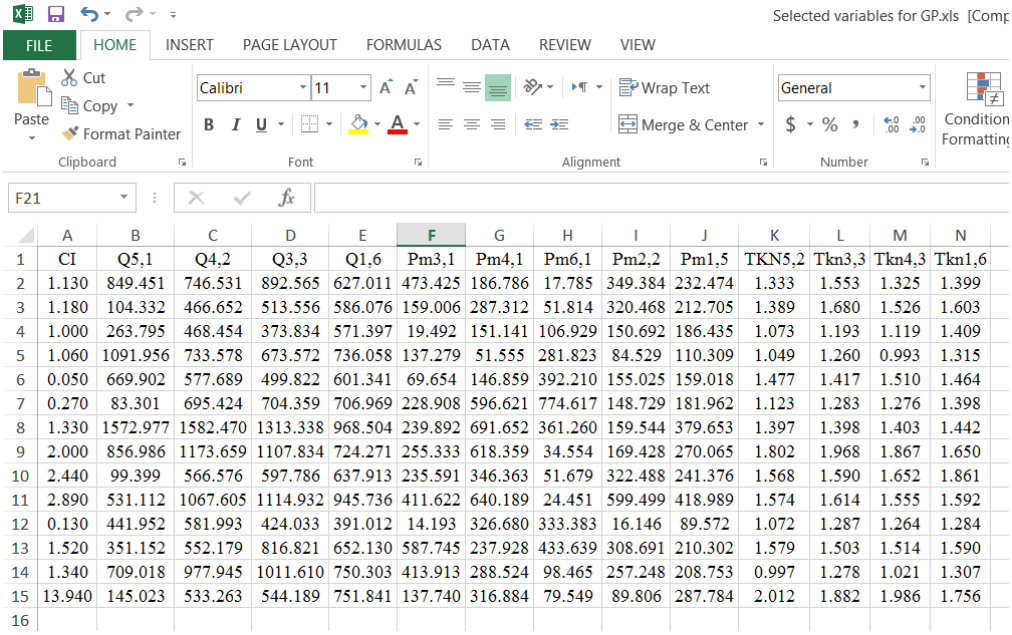


Figure C-3. Selecting Data for Training

You can get data for Discipulus training by three different methods: (1) Direct import of text files; (2) Use Notitia data import and preparation software; or (3) Import Notitia XML files directly. In this study data were imported directly from excel. User should generate an excel file from all inputs with header such as Figure C-4.



Selected variables for GP.xls [Comp

FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW

Cut Copy Format Painter Paste Clipboard

Calibri 11 Font

Wrap Text Merge & Center Alignment

General Number

Condition Formattin

F21

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	CI	Q5,1	Q4,2	Q3,3	Q1,6	Pm3,1	Pm4,1	Pm6,1	Pm2,2	Pm1,5	TKN5,2	Tkn3,3	Tkn4,3	Tkn1,6
2	1.130	849.451	746.531	892.565	627.011	473.425	186.786	17.785	349.384	232.474	1.333	1.553	1.325	1.399
3	1.180	104.332	466.652	513.556	586.076	159.006	287.312	51.814	320.468	212.705	1.389	1.680	1.526	1.603
4	1.000	263.795	468.454	373.834	571.397	19.492	151.141	106.929	150.692	186.435	1.073	1.193	1.119	1.409
5	1.060	1091.956	733.578	673.572	736.058	137.279	51.555	281.823	84.529	110.309	1.049	1.260	0.993	1.315
6	0.050	669.902	577.689	499.822	601.341	69.654	146.859	392.210	155.025	159.018	1.477	1.417	1.510	1.464
7	0.270	83.301	695.424	704.359	706.969	228.908	596.621	774.617	148.729	181.962	1.123	1.283	1.276	1.398
8	1.330	1572.977	1582.470	1313.338	968.504	239.892	691.652	361.260	159.544	379.653	1.397	1.398	1.403	1.442
9	2.000	856.986	1173.659	1107.834	724.271	255.333	618.359	34.554	169.428	270.065	1.802	1.968	1.867	1.650
10	2.440	99.399	566.576	597.786	637.913	235.591	346.363	51.679	322.488	241.376	1.568	1.590	1.652	1.861
11	2.890	531.112	1067.605	1114.932	945.736	411.622	640.189	24.451	599.499	418.989	1.574	1.614	1.555	1.592
12	0.130	441.952	581.993	424.033	391.012	14.193	326.680	333.383	16.146	89.572	1.072	1.287	1.264	1.284
13	1.520	351.152	552.179	816.821	652.130	587.745	237.928	433.639	308.691	210.302	1.579	1.503	1.514	1.590
14	1.340	709.018	977.945	1011.610	750.303	413.913	288.524	98.465	257.248	208.753	0.997	1.278	1.021	1.307
15	13.940	145.023	533.263	544.189	751.841	137.740	316.884	79.549	89.806	287.784	2.012	1.882	1.986	1.756
16														

Figure C-4. Set of Inputs for GP software

3. Identify the Problem Type and Fitness Function.

The third window in the project setup wizard is the Select Problem Type and Fitness Function Window. This window automatically detects which problem types are appropriate for your target output and also detects which fitness functions are part of the license you acquired. Thus, you may find Classification problem types greyed out if your target output has many different values. And, if you own the Professional version of Discipulus (with no add-ons), the advanced fitness functions (ranking and logistic) are not available. Figure C-5 shows the window that lets you choose the problem type and the fitness function.

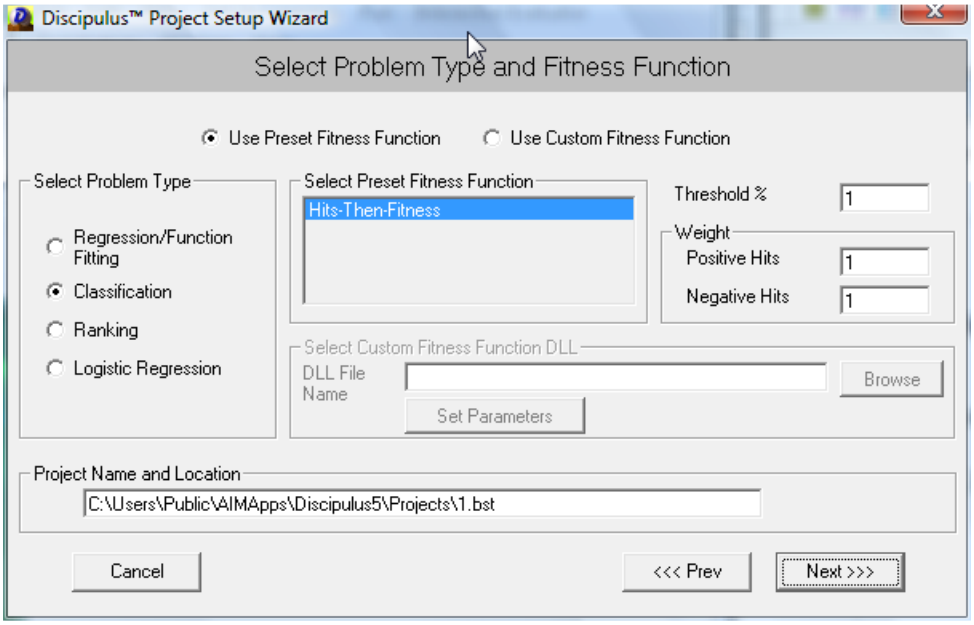


Figure C-5. The Problem Type and Fitness Function Window of the Project Wizard

4. Start the Project.

The fourth window in the Project Wizard is the "Customize Parameters and Start Project" window. To start the project, click on the "GO" button like in Figure C-6. Discipulus is entirely configured. As the project proceeds, Discipulus will intelligently adjust its own configuration.

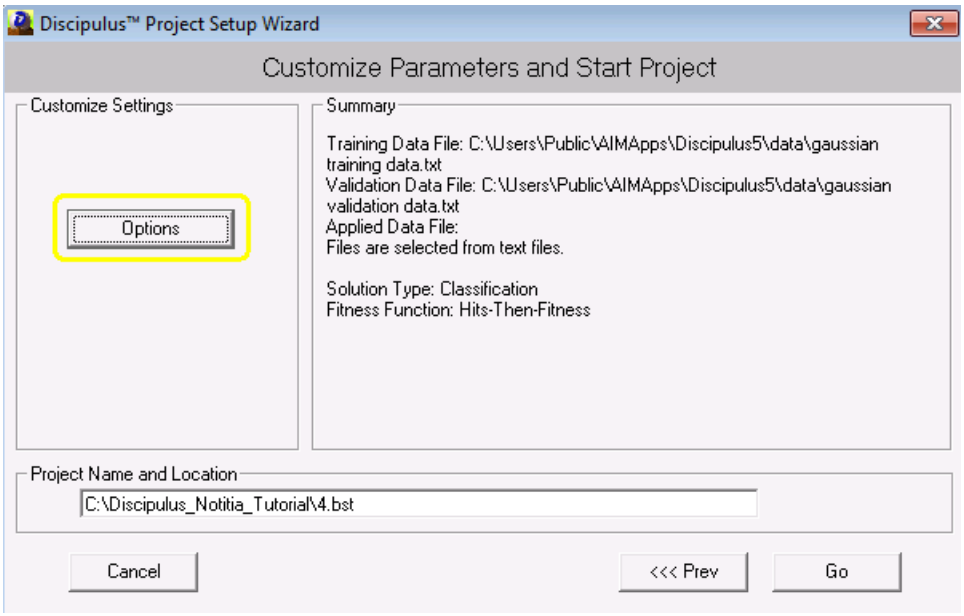


Figure C-6. Click "GO" to start your project.

The default settings for Discipulus start a project in "stepping" mode. Discipulus starts with very short runs and then increases the length of the runs as the project continues. In other words, by default, Discipulus handles run termination for you. If you want, you can set the run termination criterion manually in the Advanced Options window shown in Figure C-7. What you set will applied to all runs in the project.

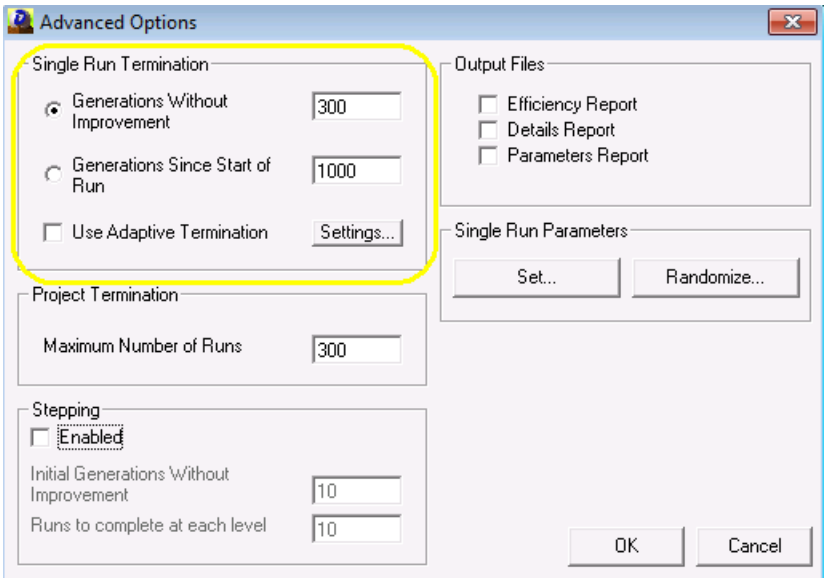


Figure C-7. The Advanced Options Window

In the highlighted area, you may select a run termination of either "generations since start" (that is, since the start of the run) or "generations without improvement" (that is, generations since there has been an improvement in the best program in the run.) You can get to the Advanced Options Window in two different ways from the main menu and from the Project Wizard:

Method 1. From the Set Up Learning Menu, select, Options.

Method 2. The final window in the Project Wizard is the Customize Parameters and Start Run window. Select Options from that window as shown in Figure C-8.

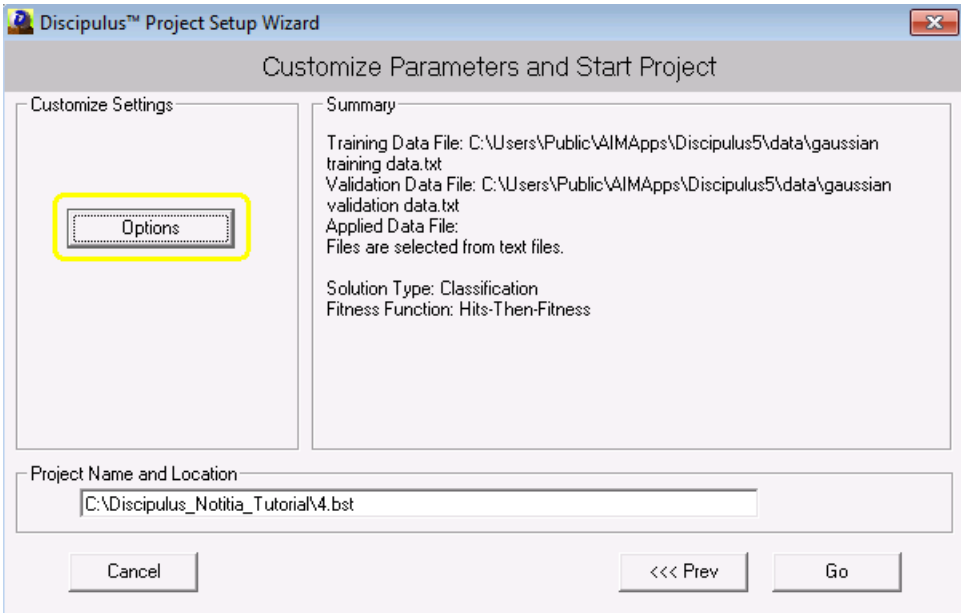


Figure C-8. The Customize Parameters and Start Run Window in the Project Wizard. Options Button Highlighted.

You can terminate a Discipulus Project manually or automatically.

Manual Project Termination

You can always stop a project manually by clicking on the "Finish Project" button on the Monitor Project Window. Alternatively, you may click on the Finish tool on the toolbar shown in Figure C-9.

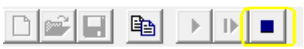


Figure C-9. The Discipulus Toolbar with the Finish Run Tool Highlighted

By default, Discipulus runs in "stepping" mode. It starts with a series of very short runs and then increases the length of the runs as the project proceeds. In this mode, a project may only be terminated manually.

Automatic Project Termination

You may elect to terminate your project after a fixed number of runs. To do so, go to the Advanced Options Window, make sure "stepping is unchecked, and enter a value for Maximum Number of Runs as shown in Figure C-10.

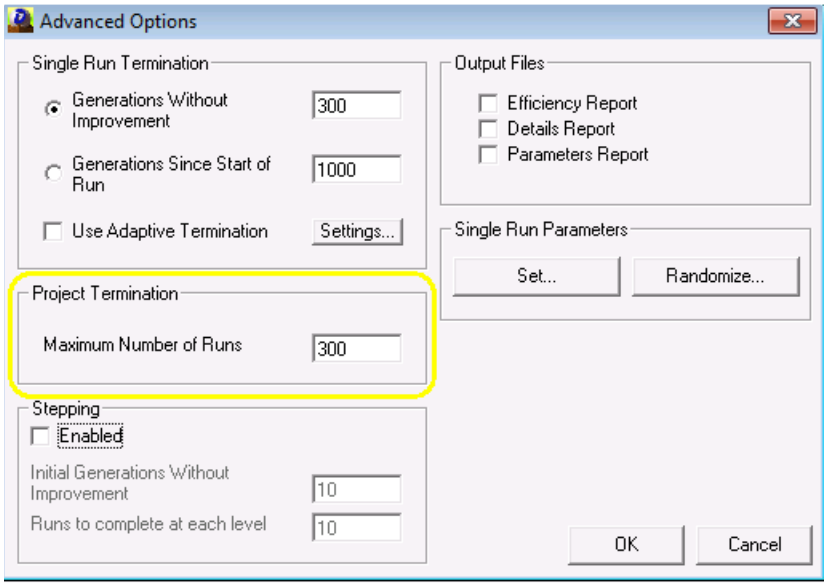


Figure C-10. Using the Advanced Options Window to set a Maximum Number of Runs in a Project to 300

Can I View the Predicted Outputs of an Evolved Best Program or a Best Team?

Yes. After a project is over, Discipulus saves the thirty best evolved program models and the five best team models and automatically brings up the Reports Window. To see the outputs of one of your best program models, select that model in the Best Programs Tab of the Reports Window. Then Click the View Results button as shown in Figure 31.

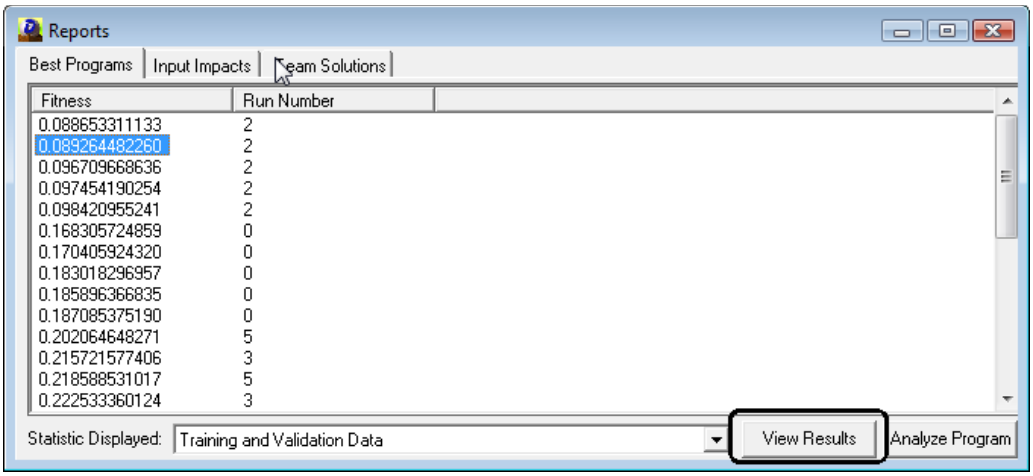


Figure C-11. The Second Best Evolved Program Model Is Selected in the Best Programs Tab. View Results Sends that Program's Outputs to the Data Window.

When you click the View Results button, the Data Window will open. In Data Window chart view, the output of the program you just selected will be shown in the "Selected Program" data series. In the Data Window spreadsheet view, the output of the program you just selected will be shown in the "Selected Program" column or columns as shown in Figure C-12.

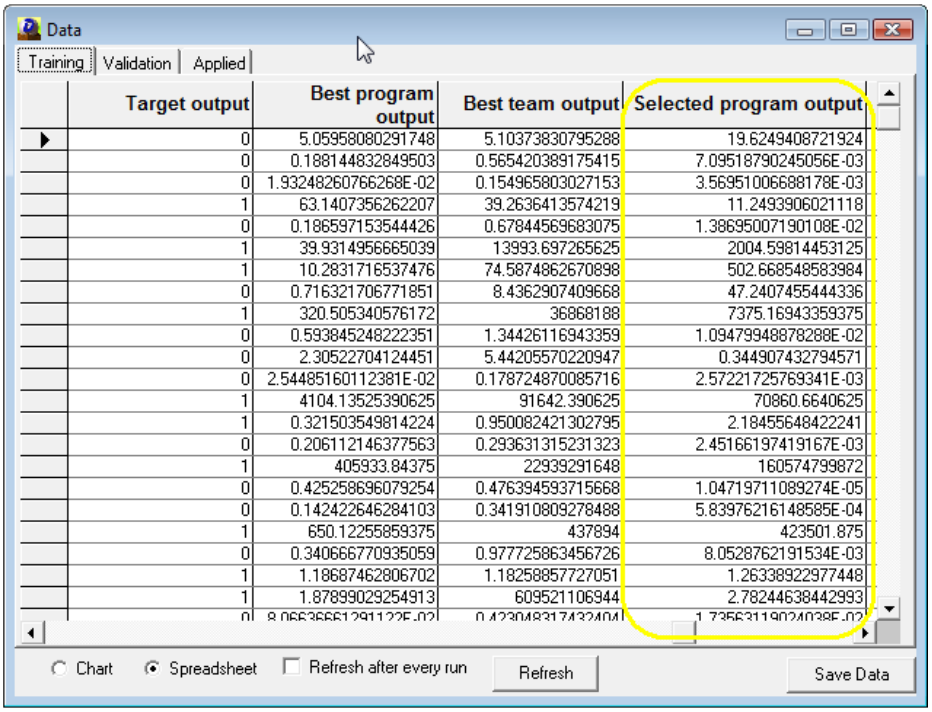


Figure C-12. The Output of a Selected Program Model is Shown in the Selected Program Output Column of the Data Window (Highlighted)

You can view the outputs of a selected Team Model in a similar way. Figure C-13 shows the Team Solutions Tab of the Reports Window with the best five member team selected.

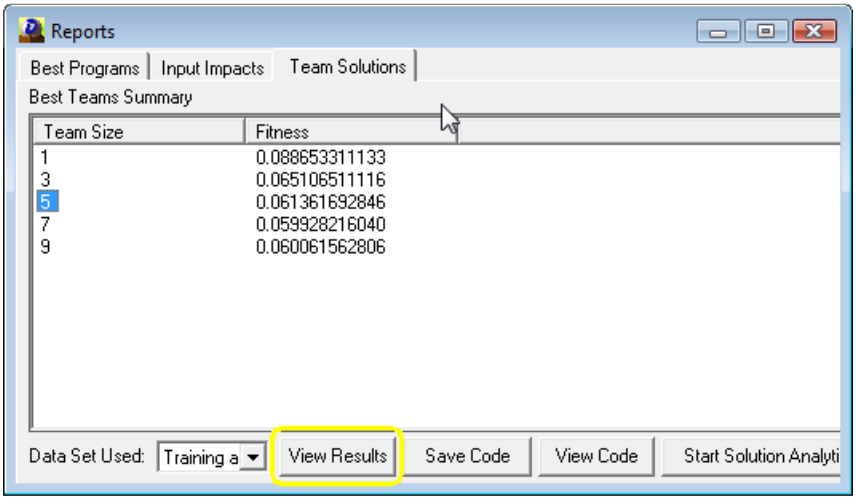


Figure C-13. Best Five Member Team Selected in the Team Solutions Tab of the Reports Window

If you click on View Results, the output of the selected team is sent to the Data Window in the "Selected Program" data series. Thus, in Data Windowchart view, you would see that selected program data as a line on the chart labeled "Selected Program." In spreadsheet view, you would see the output of that selected team as a column of outputs in a spreadsheet. The spreadsheet view is shown above in Figure C-12 with the Selected Program column highlighted.

How Do I View an Evolved Best Program Created by Discipulus?

When a Discipulus project finishes, Discipulus automatically opens a Reports Window. The Best Program Tab provides a list of the 30 best programs from that project. To save one of those programs, select it with your mouse and then click the Analyze Program button, as shown in Figure C-14. In that figure, the third best program of the project has been selected.

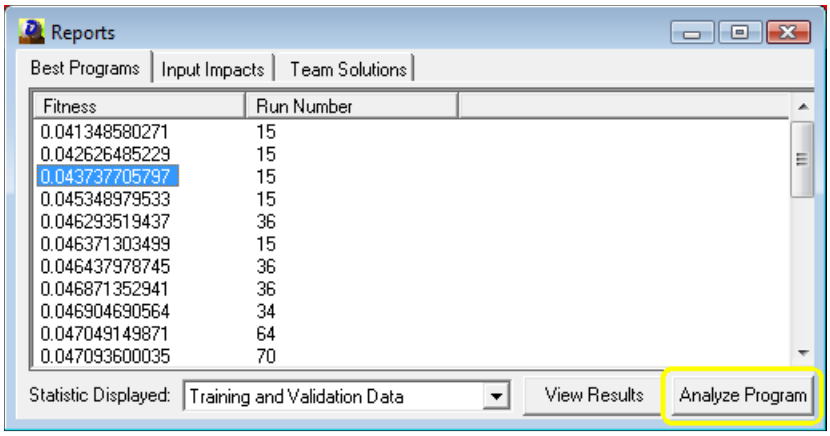


Figure C-14. Best Programs Tab Showing Third Best Program Selected and Analyze Program Button Highlighted

When you click Analyze Program, the Interactive Evaluator Window will open and the program you just selected will be displayed along with performance statistics for that program as shown in Figure C-15.

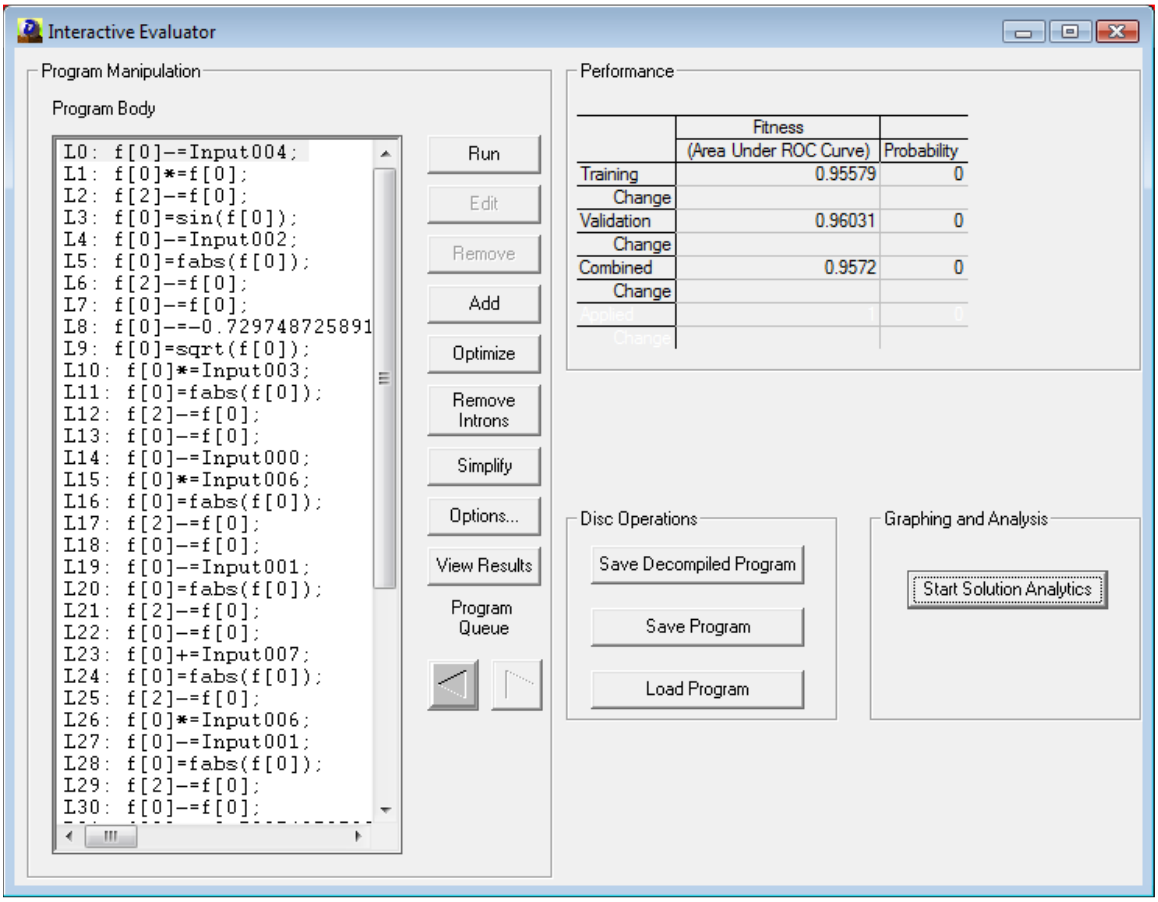


Figure C-15. Interactive Evaluator Window Displaying a Best Program

The functionality of the interactive evaluator window is documented in the chapter of the Discipulus Owner's Manual devoted to that subject.

How Do I Save an Evolved Best Program Created by Discipulus?

When a Discipulus project finishes, Discipulus automatically opens a Reports Window. The Best Program Tab provides a list of the 30 best programs from that project. To save one of those programs, select it with your mouse and then click the Analyze Program button, as shown in Figure C-16. In that figure, the third best program of the project has been selected.

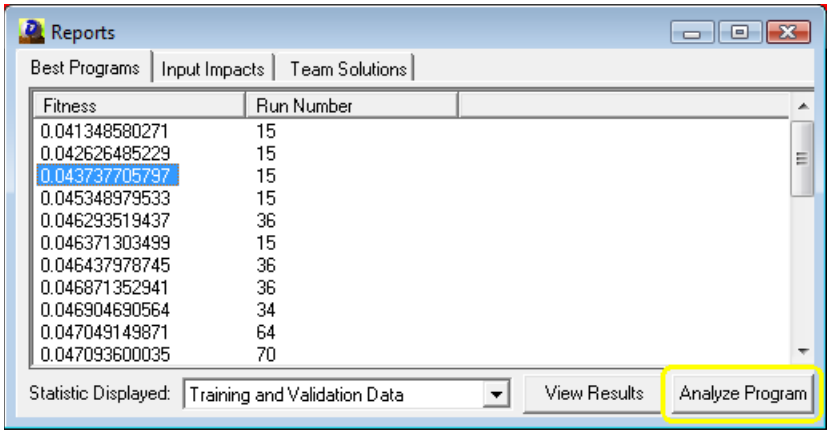


Figure C-16. Best Programs Tab Showing Third Best Program Selected and Analyze Program Button Highlighted

When you click Analyze Program, the Interactive Evaluator Window will open and the program you just selected will be displayed along with performance statistics for that program as shown in Figure C-17.

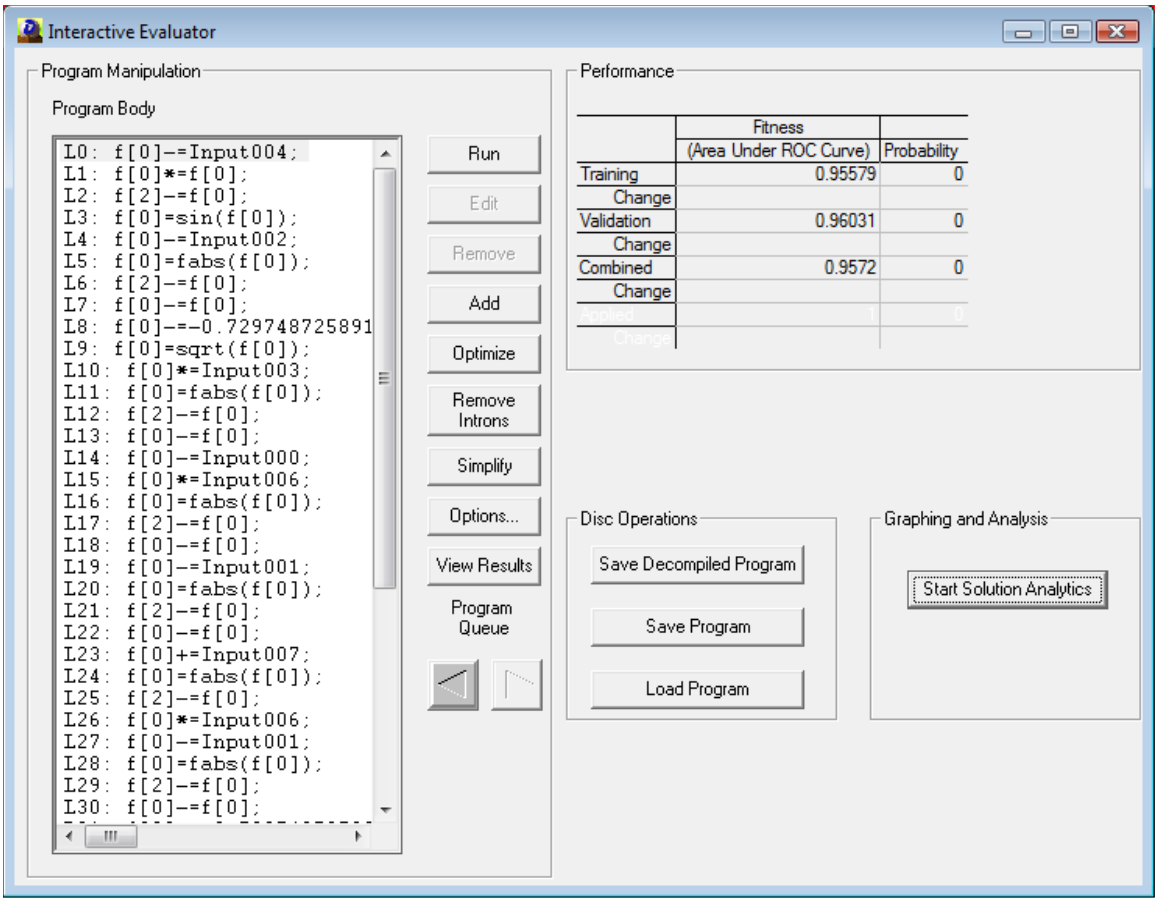


Figure C-17. Interactive Evaluator Window Displaying a Best Program

At this point, you may save your selected best program using two different methods. One method saves the selected program as object code. The other saves it in a format that lets you reload the program into Discipulus later:

Method 1--Save Program as Object Code:

To save your program as object code, click on the Save Decompiled Program button. That will bring up the window shown in Figure C-18.

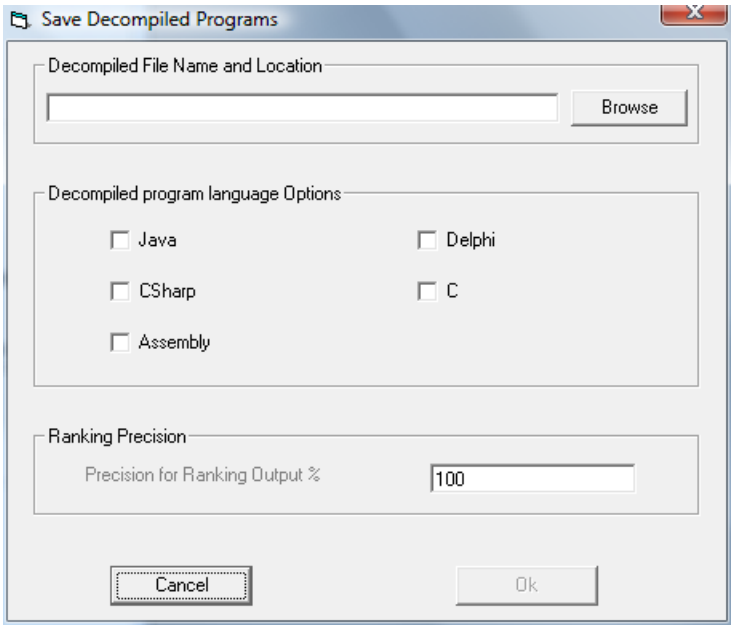


Figure C-18. Save Decompiled Program Window

Just select a computer language to save your best program in. Then select Browse to designate the folder and file name for your selected program. Discipulus automatically adds the correct extension for the particular computer language you select.

Method 2--Save Program in Reusable Format: To save your program in a format that can be loaded back into Discipulus for further use, click on the Save Program button in the Interactive Evaluator Window. This takes you to a Windows browser in which you may designate the folder and file name for the

Addition Instruction Group

The addition instruction group includes three instructions, which are discussed in the following topics:

• Add two registers: See FADD ST(0), ST(%r)

• Add two registers: See FADD ST(%r), ST(0)

• Add register and input or register and constant: See FADD [ESD+%d1]

FADD ST(0), ST(%r)

This instruction adds any one of the temporary computation variables (f[n]) to the value in f[0] and puts the sum into f[0].

C Code Description

This operator is equivalent to the following C pseudo code:

f[0]=f[0]+f[n] (or f[0]+=f[n]);

Where f[0] represents the first temporary computation variable and where f[n] represents any of the temporary computation variables you have configured Discipulus to use. The value of n is variable and is set during evolution.

Assembler Description

This instruction adds the value in the top of the FPU stack (ST(0)) to the value in variable FPU register designated as (%r). It places the sum into the top of the stack (ST(0)). The value in %r is variable and is set during evolution.

FADD ST(%r), ST(0)

This instruction adds any one of the temporary computation variables (f[n]) to the value in f[0] and puts the sum into f[n].

C Code Description

This instruction is equivalent to the following C pseudo code:

f[n]=f[0]+f[n] (or f[n]+=f[0]);

Where f[0] represents the first temporary computation variable and where f[n] represents any one of the temporary computation variables you have configured Discipulus to use. The value of n is variable and is set during evolution.

Assembler Description

This instruction adds the value in the top of the FPU stack (ST(0)) to the value in variable FPU register designated as (%r). It places the sum into the variable FPU register designated as (%r). The value in %r is variable and is set during evolution.

FADD [ESD+%d1]

This instruction will put two different operators into your evolved programs:

• The first adds f[0] to one of the inputs from your data file and places the result into f[0];

• The second adds f[0] to one of the constants from the Terminal Set and places the result into f[0].

C Code Description

The two operators referred to above are equivalent to the following lines of C pseudo code in evolved programs:

f[0]=f[0]+input (or f[0]+=input)

f[0]=f[0]+constant (or f[0]+=constant)

f[0] is, of course, the temporary calculation register. The input will show up in your evolved programs as Input001, Input002 . . . The constant will show up as a real valued constant, such as 9.1234567. During evolution, an input can be changed by the mutation operator to a constant and vice versa. Similarly, which input or constant is referenced in this operator may be changed by the mutation operator.

Assembler Description

This instruction adds the value in the top of the FPU stack (ST(0)) to the value of one of the inputs in your training data set or one of the constants. It places the sum into the top of the stack (ST(0)). The value in %d1 is variable (that is, which variable or which constant) and is set during evolution.

Arithmetic Instruction Group

The Arithmetic Instruction Group contains four instructions that are described in the following topics:

• Absolute Value. See FABS

• Change Sign. See FCHS

• Scaling. See FSCALE

• Square root, See FSQRT

FABS

This instruction takes the absolute value of f[0] and places the result into f[0].

C Code Description

It is equivalent to this C pseudo code:

f[0]=ABS(f[0]);

Assembler Description

Takes the absolute value of the top of the FPU stack (ST(0)). It places that absolute value back into the top of the stack (ST(0)).

FCHS

This instruction changes the sign of f[0] and places the result into f[0].

C Code Description

This instruction is equivalent to this C pseudo code:

f[0]=–(f[0]);

Assembler Description

Changes the sign of the value in the top of the stack register, ST(0).

FSCALE

This instruction multiplies f[0] by two raised to the power, f[1]. It then places the result back into f[0].

C Code Description

It is equivalent to this pseudo code:

f[0]=f[0]*(2^f[1]);

Assembler Description

Calculates ST(0)*2^ST(1) and places the result into ST(0).

FSQRT

This instruction takes the square root of f[0] and places the result into f[0].

C Code Description

This instruction is equivalent to the following C pseudo code:

f[0]=SQRT(f[0]);

Assembler Description

Takes the square root of ST(0) and places the result into ST(0).

Comparison Instruction Group

The comparison instruction group contains only one instruction, which compares the values in two floating point registers. See FCOMI ST(0), ST(%r).

FCOMI ST(0), ST(%r)

Compares the values in f[0] and f[n]. If f[0] is less than f[n], it sets the temporary variable, cflag to 1, otherwise, it set cflag to 0.

C Code Description

This instruction is equivalent to the following C pseudo code:

cflag=(f[0]<f[n]);

Where cflag is a Boolean variable that can have only the values of 0 or 1 and where f[n] is the value in one of the n temporary computation variables.

Assembler Description

Compares the contents of register ST(0) and ST(n) and sets the status flags ZF, PF, and CF in the EFLAGS register according to the results.

Condition Instruction Group

The conditional instructions work with the Comparison Instruction Group. The Comparison Instructions set the value of cflag by comparing the values in f[0] and f[1]. Then the Condition Instructions use the value in cflag to decide whether or not to take one of two steps:

• Move the value in f[n] to f[0]; or

• Jump over one Instruction Block.

The following topics describe the Conditional Instructions you may include in Discipulus programs:

• Conditional copy of value from one register to f[0]: See FCMOVB ST(0), ST(%r)

• Conditional copy of value from f[0] to another register: See FCMOVNB ST(0), ST(%r)

• Conditional jump of an Instruction Block if cflag = 1: See JB EPI+6

• Conditional jump of an Instruction Block if cflag = 0; JNB EPI+6

FCMOVB ST(0), ST(%r)

This instruction moves the value in f[n] to f[0] if the conditional flag (cflag) is equal to 1. (The conditional flag is set by the Comparison Group instructions.)

C Code Description

This instruction is equivalent to the following C pseudo code:

if (cflag) f[0] = f[n];

Assembler Description

Tests the CF status flag and moves the source operand (ST(n)) to the destination operand (ST(0)), if CF=1.

FCMOVNB ST(0), ST(%r)

This instruction moves the value in f[n] to f[0] if the conditional flag (cflag) is equal to 0. (The conditional flag is set by the Comparison Group instructions.)

C Code Description

Equivalent C pseudo code is:

if (!cflag) f[0] = f[n];

Assembler Description

Tests the CF status flag and moves the source operand (ST(n)) to the destination operand (ST(0)), if CF=0.

JB EPI+6

This instruction causes the program to skip execution of the next Instruction Block if the conditional flag (cflag) equals 1. (The conditional flag is set by the Comparison Group instructions.)

C Code Description

A C code example follows. This code tests whether cflag=1. If it does, the program skips over line 12:

11: if (cflag) goto 13;

12: f[0]+=1.234567;

13: f[0]*=f[0];

Assembler Description

Tests the CF status flag and jumps program execution by 6 bytes if

CF=1.

JNB EPI+6

This instruction causes the program to skip execution of the next Instruction Block if the conditional flag (cflag) equals 0. (The conditional flag is set by the Comparison Group instructions.)

C Code Description

A C code example follows. This code tests whether cflag=0. If it does, the program skips over line 12.

11: if (!cflag) goto 13;

12: f[0]+=1.234567;

13: f[0]*=f[0]

Assembler Description

Tests the CF status flag and jumps program execution by 6 bytes if

CF=0.

Data Transfer Instruction Group

The Data Transfer Instructions move values around without changing the values. The one such instruction implemented in Discipulus exchanges values between f[0] and another register. See FXCH ST(%r)

FXCH ST(%r)

The FXCH instruction swaps the values in f[0] and f[n]. This is an important instruction in Register Machine configurations because it allows the system to move values to and from the higher f[n] variables for temporary storage while other calculations are performed in f[0].

C Code Description

The FLD instructions are equivalent to the following C pseudo code:

tmp=f[0];

f[0]=f[n];

f[n]=tmp;

Assembler Description

Swap the values in ST(0) and ST(n).

Division Instruction Group

The Division Instruction Group includes four instructions that are detailed in the following topics:

• Divide one register by another; place the result in f[0]: See FDIV ST(0), ST(%r)

• Divide one register by another; place the result in f[n]; See FDIV ST(%r), ST(0)

• Calculate a remainder; See FPREM

• Divide f[0] by either a constant or an input value: See FDIV [ESD+%d1]

FDIV ST(0), ST(%r)

This instruction divides one of the temporary computation variables (f[0]) by the value in f[n] and puts the difference into f[0].

C Code Description

This operator is equivalent to the following C pseudo code:

f[0]=f[0]/f[n] (or f[0]/=f[n]);

Where f[0] represents the first temporary computation variable and where f[n] represents any of the temporary computation variables you have configured Discipulus to use. The value of n is variable and is set during evolution.

Assembler Description

This instruction divides the value in the top of the FPU stack (ST(0)) by the value in variable FPU register designated as (%r). It places the difference into the top of the stack (ST(0)). The value in %r is variable and is set during evolution.

FDIV ST(%r), ST(0)

This instruction divides one of the temporary computation variables (f[n]) by the value in f[0] and puts the difference into f[n].

C Code Description

This instruction is equivalent to the following C pseudo code:

f[n]=f[n]/f[0] (or f[n]/=f[0]);

Where f[0] represents the first temporary computation variable and where f[n] represents any one of the temporary computation variables you have configured Discipulus to use. The value of n is variable and is set during evolution.

Assembler Description

This instruction divides the value in the top of the FPU stack (ST(0)) by the value in variable FPU register designated as (%r). It places the result into the variable FPU register designated as (%r). The value in %r is variable and is set during evolution.

FPREM

This operator causes an evolved program calculate the remainder left when f[0] is divided by f[1] and to place the result into f[0]. This instruction is useful for periodic data.

C Code Description

This instruction is equivalent to the following C pseudo code:

f[0]=f[0]– ((int)(f[0]/f[1])*f[1]);

f[0] and f[1] are, of course, temporary calculation variables.

Assembler Description

Computes the remainder obtained from dividing the value in the ST(0) register (the dividend) by the value in the ST(1) register (the divisor or modulus), and stores the result in ST(0). The remainder represents the following value:

Remainder = ST(0) - (Q * ST(1))

Here, Q is an integer value that is obtained by truncating the real number quotient of [ST(0) /ST(1)] toward zero. The sign of the remainder is the same as the sign of the dividend. The magnitude of the remainder is less than that of the modulus.

FDIV [ESD+%d1]

This instruction will put two different types of code into your evolved programs:

• The first divides f[0] by one of the inputs from your data file and places the result into f[0];

• The second divides f[0] by one of the constants from the Terminal Set and places the result into f[0].

C Code Description

This operator causes an evolved program to include both of the following lines of C pseudo code in evolved programs:

f[0]=f[0]–input (or f[0]–=input); and

f[0]=f[0]–constant (or f[0]–=constant);

f[0] is, of course, the temporary calculation register. The input will show up in your evolved programs as Input001, Input002. . . The constant will show up as a real valued constant, such as 9.1234567. During evolution, an input can be changed by the mutation operator to a constant and vice versa. Similarly, which input or constant is referenced in this operator may be changed by the mutation operator.

Assembler Description

This instruction subtracts the value in one of the inputs in your training data set or one of the constants, from the value in the top of the FPU stack (ST(0)). It places the difference into the top of the stack (ST(0)). The value in %d1 represents which value is subtracted (that is, which variable or which constant) and is set during evolution.

Exponential Instruction Group

This instruction group implements only one instruction, **F2XM1**. The F2XM1 instruction calculates two raised to the f[0] power, minus one and puts the result into f[0].

C Code Description

This operator is equivalent to the following C pseudo code:

if (fabs(f[0])<1) f[0]=pow(2,f[0])-1;

Where f[0] represents the first temporary computation variable.

Assembler Description

Calculates the exponential value of 2 to the power of the source operand minus 1. The source operand is located in register ST(0) and the result is also stored in ST(0). The value of the source operand must lie in the range –1.0 to +1.0. If the source value is outside this range, the result is undefined.

Multiplication Instruction Group

The four multiplication instructions implemented in Discipulus are discussed in the following topics:

• Multiply two registers and place the result in f[0]. See FMUL ST(0), ST(%r)

• Multiply two registers and place the result in f[n]. See FMUL ST(%r), ST(0)

• Multiply a register by an input or a constant. See FMUL [ESD+%d1]

FMUL ST(0), ST(%r)

This instruction multiplies one of the temporary computation variables (f[n]) and f[0] and puts the product into f[0].

C Code Description

This operator is equivalent to the following C pseudo code:

f[0]=f[0]*f[n] (or f[0]*=f[n]);

Where f[0] represents the first temporary computation variable and where f[n] represents any of the temporary computation variables you have configured Discipulus to use. The value of n is variable and is set during evolution.

Assembler Description

This instruction multiplies the value in the top of the FPU stack (ST(0)) and the value in variable FPU register designated as (%r). It places the product into the top of the stack (ST(0)). The value in %r is variable and is set during evolution.

FMUL ST(%r), ST(0)

This instruction multiplies the values in f[0] and f[n] together and places the results in f[n].

C Code Description

This instruction is equivalent to the following C pseudocode:

f[n]=f[0]*f[n] (or f[n]*=f[0]);

Where f[0] represents the first temporary computation variable and where f[n] represents any one of the temporary computation variables you have configured Discipulus to use. The value of n is variable and is set during evolution.

Assembler Description

This instruction multiplies the value in the top of the FPU stack (ST(0)) and the value in variable FPU register designated as (%r). It places the product into the variable FPU register designated as (%r). The value in %r is variable and is set during evolution.

FMUL [ESD+%d1]

This instruction will put two related operators into your evolved programs:

• The first multiplies f[0] and one of the inputs from your data file and places the result into f[0];

• The second multiplies f[0] and one of the constants from the Terminal Set and places the result into f[0].

C Code Description

The two related operators referred to above are equivalent to the following lines (one at a time) of C pseudocode in evolved programs:

f[0]=f[0]*input (or f[0]*=input).

f[0]=f[0]*constant (or f[0]*=constant).

f[0] is, of course, the temporary calculation register. The input will show up in your evolved programs as Input001, Input002. . . . etc. Or, if you name the input columns and use Notitia to import the data to Discipulus, your input names will appear in the evolved programs. The constant will show up as a real valued constant, such as 9.1234567. During evolution, an input can be changed by the mutation operator to a constant and vice versa. Similarly, which input or constant is referenced in this operator may be changed by the mutation operator.

Assembler Description

This instruction multiplies the value in one of the inputs in your training data or one of the constants, to the value in the top of the FPU stack (ST(0)). It places the product into the top of the stack (ST(0)). The value in %d1 represents which value is subtracted (that is, which variable or which constant) and is set during evolution.

Rotate Stack Instruction Group

FDECSTP

This instruction decrements the FPU stack pointer by 1. It makes no changes to the contents of the registers.

FINCSTP

This instruction increments the FPU stack pointer by 1. It makes no changes to the contents of the registers.

Subtraction Instruction Group

The three subtraction instructions implemented by Disciples are discussed in the following topics:

• Subtract two registers and put the result in f[0]. See FSUB ST(0), ST(%r)

• Subtract two registers and put the result in f[n]. See FSUB ST(%r), ST(0)

• Subtract an input or a constant from a register. See FSUB [ESD+%d1]

FSUB ST(0), ST(%r)

This instruction subtracts one of the temporary computation variables (f[n]) from the value in f[0] and puts the difference into f[0].

C Code Description

This operator is equivalent to the following C pseudocode:

f[0]=f[0]–f[n] (or f[0]–=f[n]);

Where f[0] represents the first temporary computation variable and where f[n] represents any of the temporary computation variables you have configured Discipulus to use. The value of n is variable and is set during evolution.

Assembler Description

This instruction subtracts the value in the top of the FPU stack (ST(0)) from the value in variable FPU register designated as (%r). It places the difference into the top of the stack (ST(0)). The value in %r is variable and is set during evolution.

FSUB ST(%r), ST(0)

This instruction subtracts f[0] from f[n] and places the result into f[n].

C Code Description

This instruction is equivalent to the following C pseudocode:

f[n]=f[n]–f[0] (or f[n]–=f[0]);

Where f[0] represents the first temporary computation variable and where f[n] represents any one of the temporary computation variables you have configured Discipulus to use. The value of n is variable and is set during evolution.

Assembler Description

This instruction subtracts the value in the top of the FPU stack (ST(0)) from the value in variable FPU register designated as (%r). It places the difference into the variable FPU register designated as (%r). The value in %r is variable and is set during evolution.

FSUB [ESD+%d1]

This instruction will put two related operators into your evolved programs:

• The first subtracts one of the inputs from your data file from f[0] and places the result into f[0];

• The second subtracts one of the constants from the Terminal Set from f[0] and places the result into f[0].

C Code Description

The two related operators referred to above are equivalent to the following lines of C pseudocode in evolved programs:

f[0]=f[0]–input (or f[0]–=input).

f[0]=f[0]–constant (or f[0]–=constant).

f[0] is, of course, the temporary calculation register. The input will show up in your evolved programs as Input001, Input002, etc. Or, if you assigned column names for your inputs and used Notitia to import the data, your column names will be used in the evolved programs. The constant will show up as a real valued constant, such as 9.1234567. During evolution, an input can be changed by the mutation operator to a constant and vice versa. Similarly, which input or constant is referenced in this operator may be changed by the mutation operator.

Assembler Description

This instruction subtracts an input or a constant from the value in the top of the FPU stack (ST(0)). It places the result into the top of the stack (ST(0)). The value in %d1 represents which value is subtracted (that is, which variable or which constant) and is set during evolution.

Trigonometric Instruction Group

The two trigonometric functions implemented in Discipulus are discussed in the following topics:

• Cosine function. See FCOS

• Sine function. FSIN

FCOS

This instruction calculates the cosine of f[0] and puts the result into f[0].

C Code Description

This operator is equivalent to the following C pseudocode:

f[0]=cos(f[0]);

Assembler Description

Calculates the cosine of the source operand in register ST(0) and stores the result in ST(0).

FSIN

This instruction calculates the sin of f[0] and puts the result into f[0].

C Code Description

This operator is equivalent to the following C pseudocode:

f[0]=sin(f[0]);

Assembler Description

Calculates the sine of the source operand in register ST(0) and stores the result in ST(0). The source operand must be given in radians.